DSP of CT Signals

Understanding Sampling in the Frequency Domain

Relate $x_c(t)$ directly to $x[n]$

Compute CTFT of $x_s(t) = \sum_{n=-\infty}^{\infty} x_c(nT)\,\delta(t-nT)$

$$X_s(\Omega) = \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x_c(nT)\,\delta(t-nT)\,e^{-j\Omega t}\,dt$$

$$= \sum_{n=-\infty}^{\infty} x_c(nT) \int_{-\infty}^{\infty} \delta(t-nT)\,e^{-j\Omega t}\,dt$$

$$= \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\Omega nT}$$

$$= \underbrace{\sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n}}_{\text{DTFT of } x[n]}, \quad \boxed{\omega \equiv \Omega T}$$

$$= X(\omega)$$

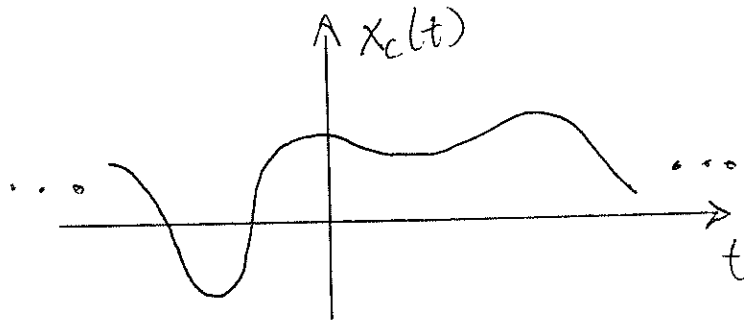
Recall $X_s(\Omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c(\Omega - k\Omega_s)$

$$\Rightarrow X(\omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c(\Omega - k\Omega_s)$$

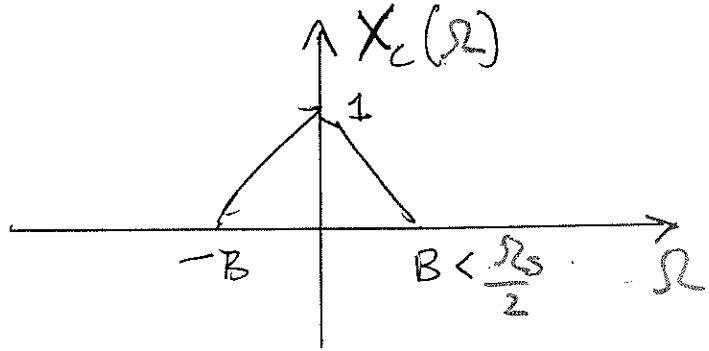
$$= \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c\left(\frac{\omega - 2\pi k}{T}\right)$$

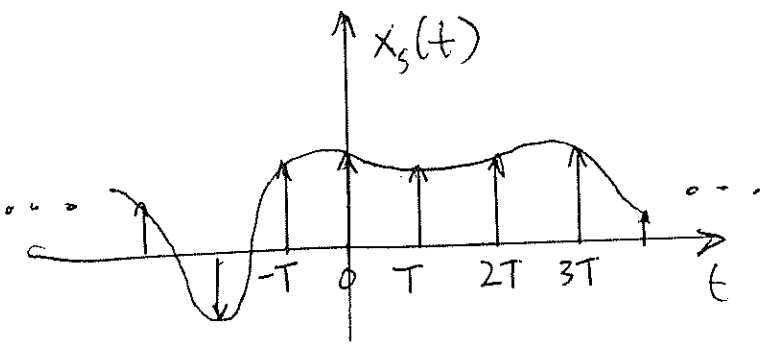
2π-periodic

SAMPLING

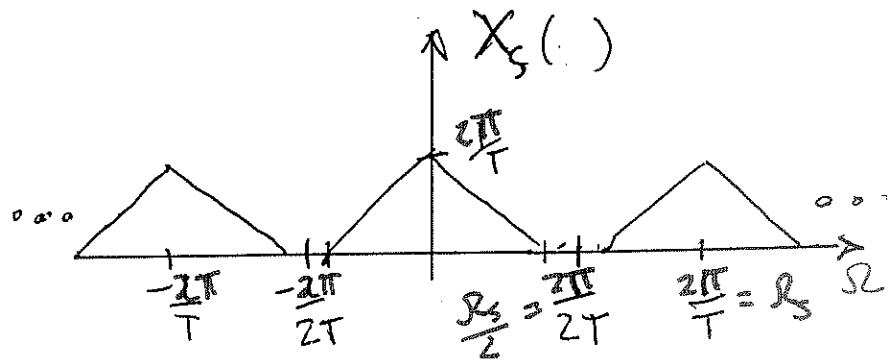


CT FT $\longleftrightarrow$

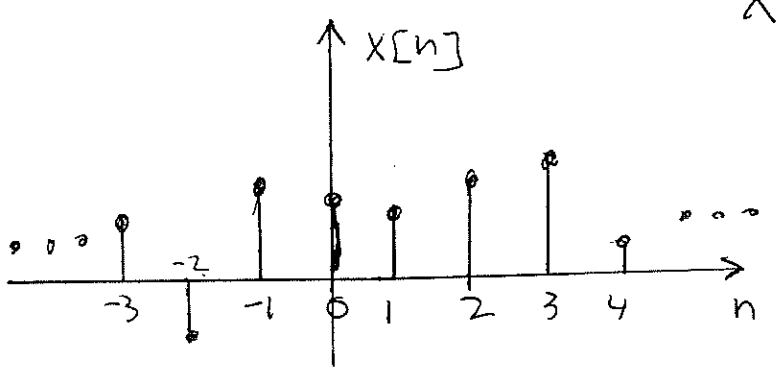




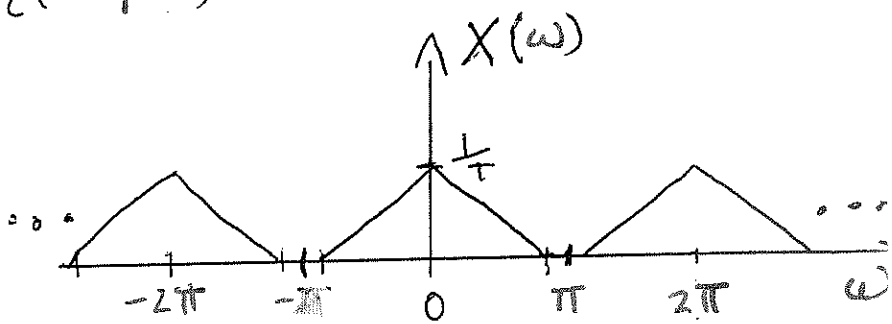
CTFT $\longleftrightarrow$



$$X(\omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c\left(\frac{\omega - 2\pi k}{T}\right)$$



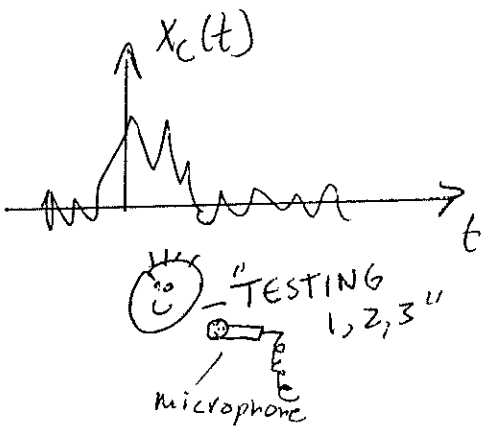
DTFT $\longleftrightarrow$

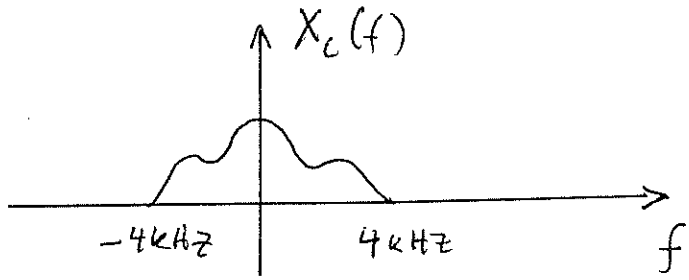


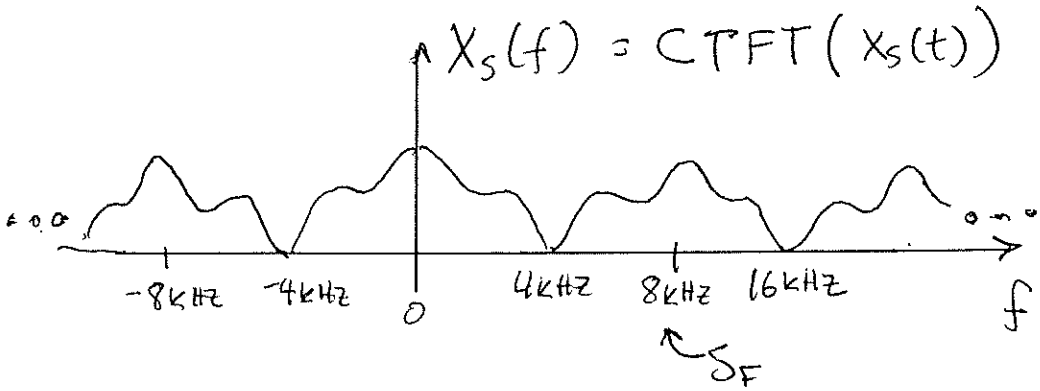
$\omega = \Omega T$, $\Omega_s = \frac{2\pi}{T}$

Example: Speech

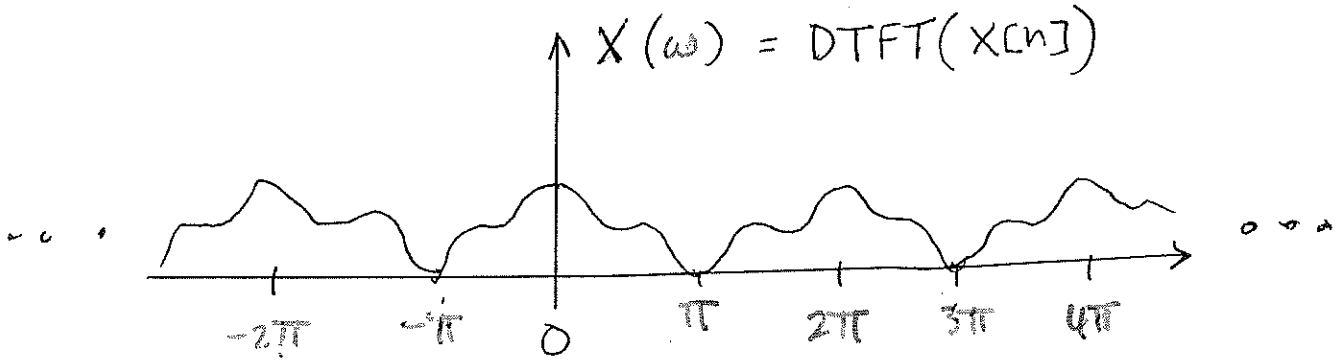
Speech is intelligable if bandlimited by a CT lowpass filter to the band ± 4kHz $\Longrightarrow$ we can sample speech as slowly as _____







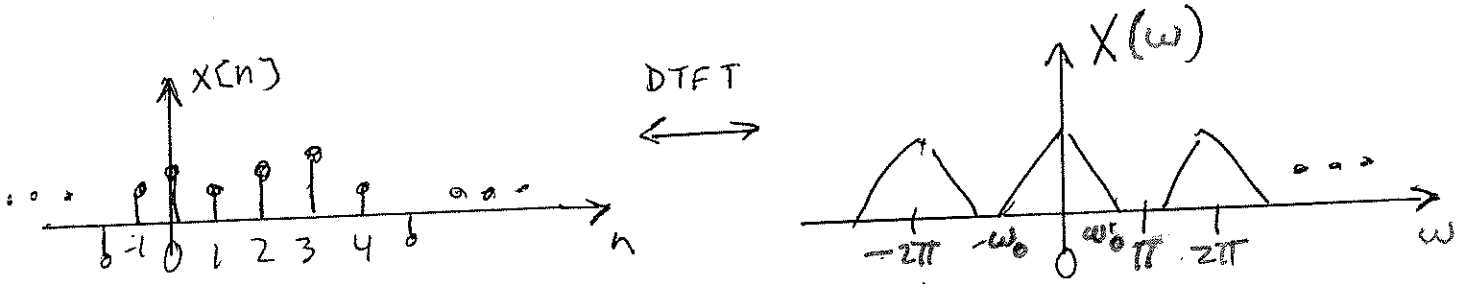
or multiply by 2π to get $\Omega = 2\pi f$



NOTE: No mention of T or Ω_s!

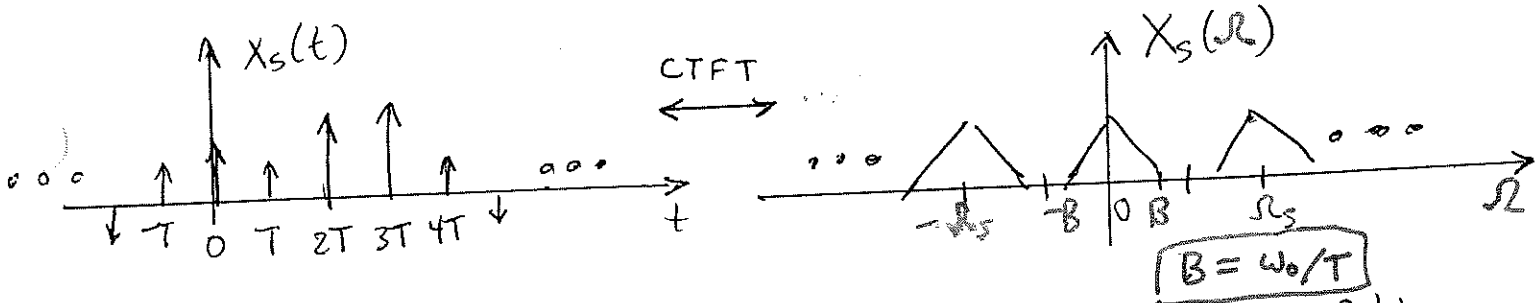
RECONSTRUCTION OF SAMPLED SIGNALS

How do we go from $x[n]$ to CT?

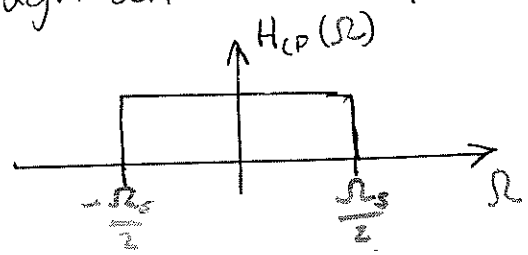


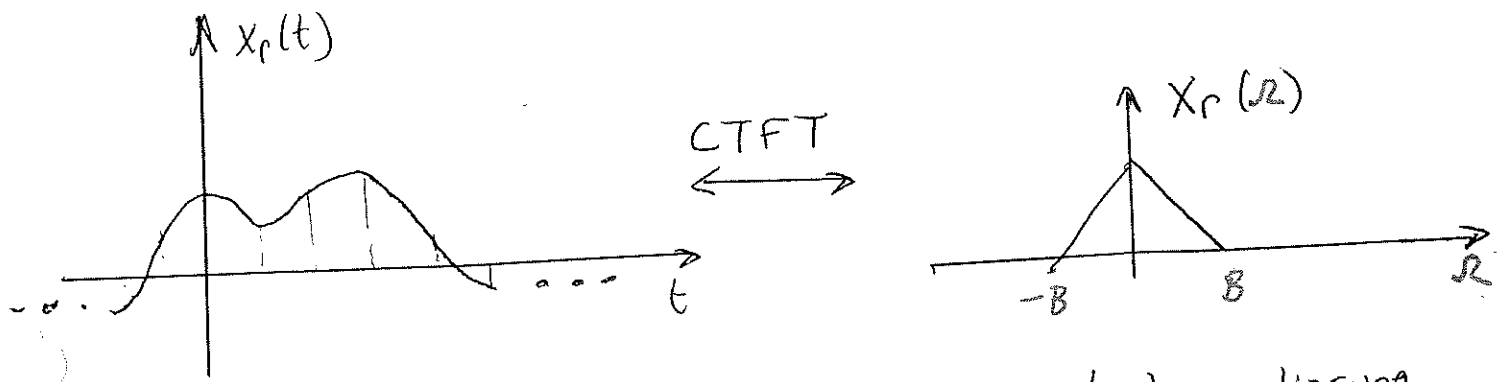
STEP 1: Place $x[n]$ into CT on an impulse train $s(t)$

$$x_s(t) = \sum_{n=-\infty}^{\infty} x[n]\,\delta(t-nT)$$



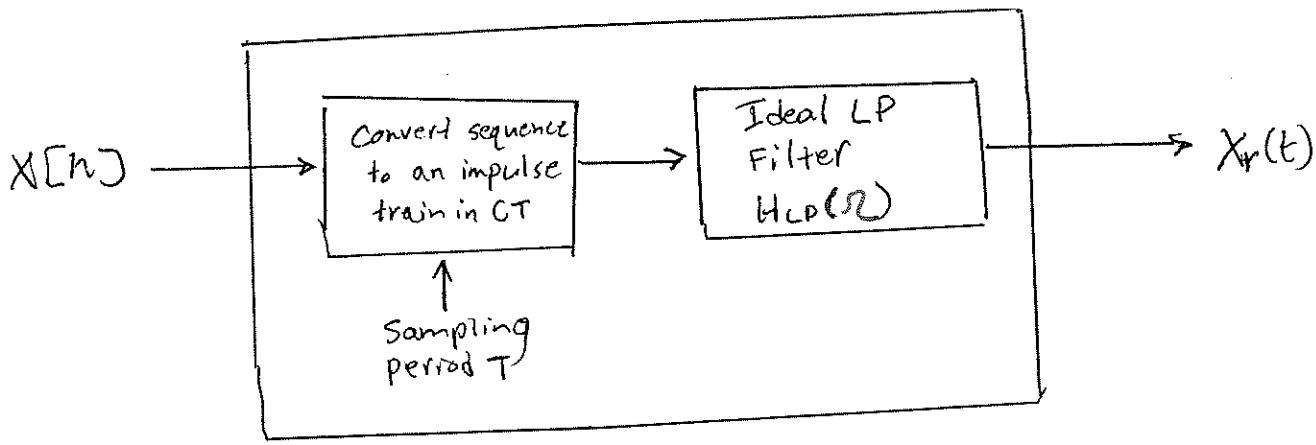
STEP 2: Pass $x_s(t)$ through an ideal lowpass filter





If we had no aliasing then $x_r(t) = x_c(t)$, where $x[n] = x_c(nT)$

Ideal Reconstruction System



In frequency domain:

1. $X_s(\Omega) = X(\Omega T)$ ← DTFT of x[n] at digital freq. $\omega = \Omega T$

2. $X_r(\Omega) = H_{LP}(\Omega) \cdot X_s(\Omega)$

$$\Longrightarrow \boxed{X_r(\Omega) = H_{LP}(\Omega) \cdot X(\Omega T)}$$

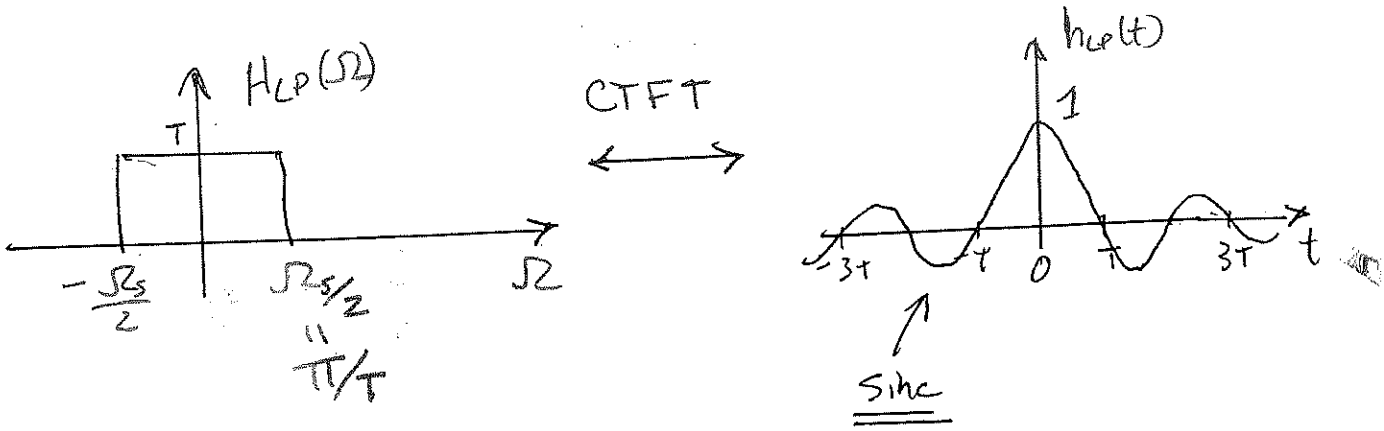
In time domain:

1. $x_s(t) = \sum_{n=-\infty}^{\infty} x[n]\,\delta(t-nT)$

2. $x_r(t) = \left(\sum_{n=-\infty}^{\infty} x[n]\,\delta(t-nT)\right) * h_{LP}(t)$ ← sinc

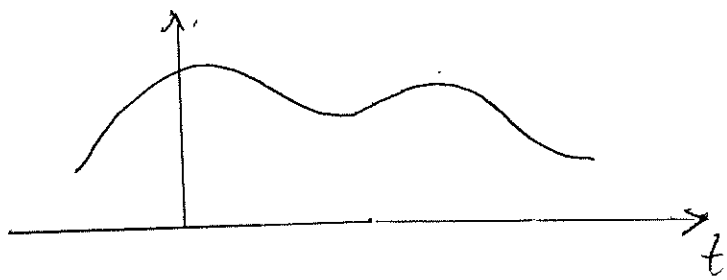
$$h_{LP}(t) = \text{sinc}\left(\frac{\pi}{T}t\right)$$

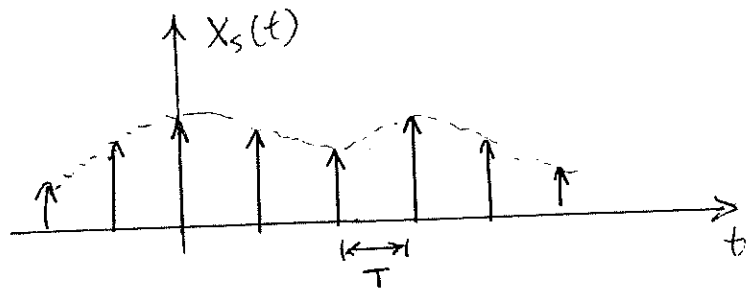
$$\boxed{x_r(t) = \sum_{n=-\infty}^{\infty} x[n]\,\text{sinc}\left(\frac{\pi}{T}(t-nT)\right)}$$ ← interpol

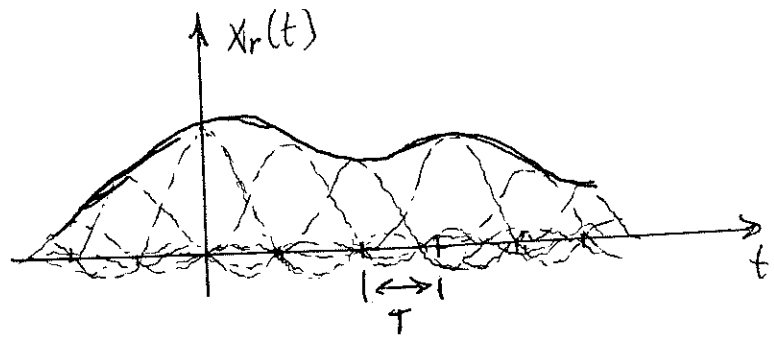


$$h_{LP}(t) = \text{sinc}\left(\frac{\pi}{T}t\right) = \frac{\sin\left(\frac{\pi}{T}t\right)}{\frac{\pi}{T}t}$$

$h_{LP}(t)$ "interpolates" the values of $x[n]$ to generate $x_r(t)$





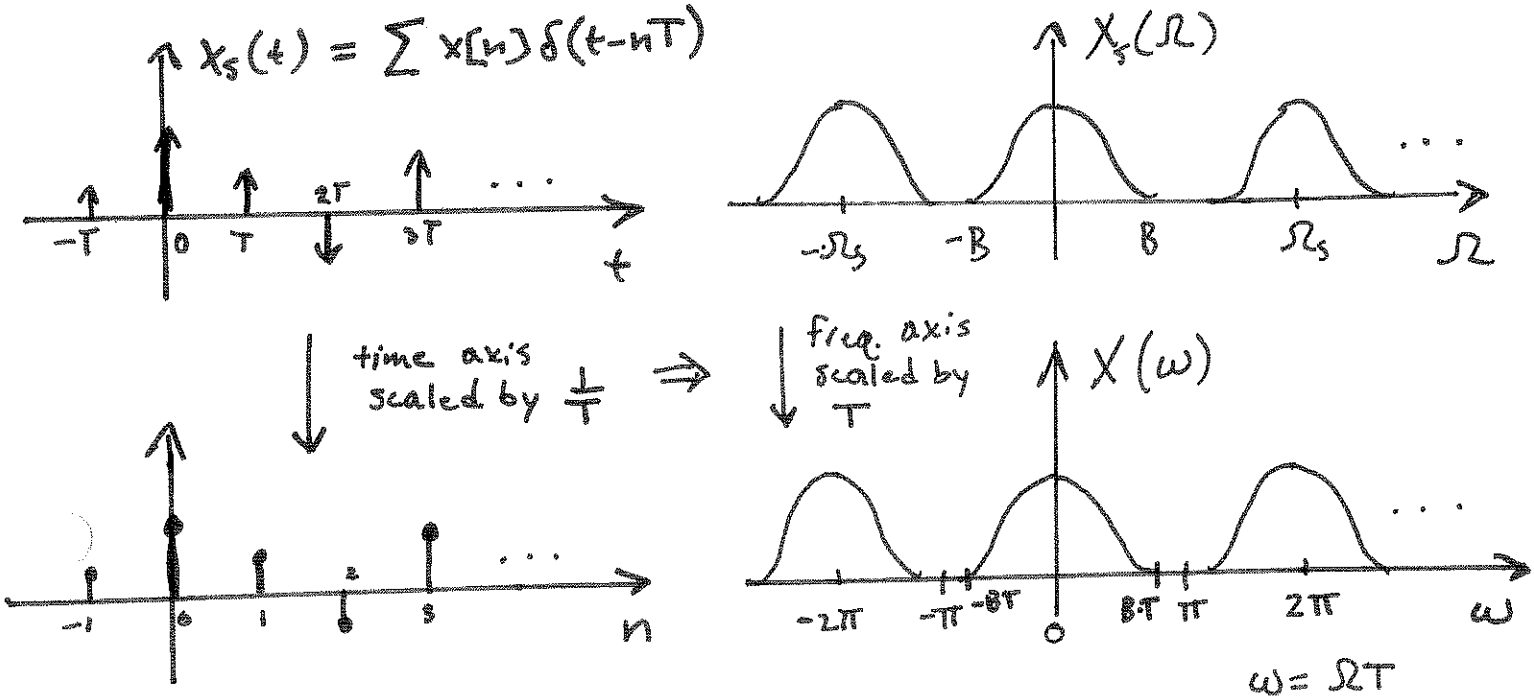


$$x_r(t) = \sum_{n=-\infty}^{\infty} x[n] \frac{\sin\left(\frac{\pi}{T}(t-nT)\right)}{\left(\frac{\pi}{T}(t-nT)\right)}$$ ← "sinc interpolator

Relating $x_s(t) = \sum x(nT)\,\delta(t-nT)$ and $x[n]$

TIME

FREQUENCY



Recall CTFT relation:

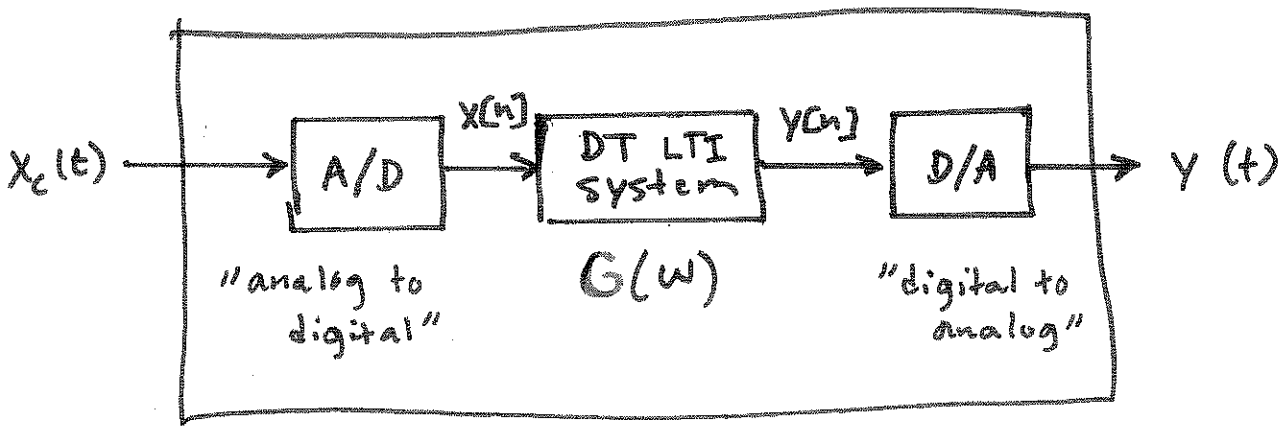
$$x(at) \longleftrightarrow \frac{1}{a} X\left(\frac{\Omega}{a}\right)$$

a scaling of time

$\frac{1}{a}$ scaling in freq.

$$\boxed{X_s(\Omega) \equiv X(\Omega \cdot T)}$$

Discrete-Time Processing of Continuous-Time Signals



DSP System

Analysis:

$$Y_c(\Omega) = H_{LP}(\Omega)\, Y(\Omega \cdot T)$$

$$\| $$

$$Y(\omega) = X(\omega) \cdot G(\omega)$$

$Y(\omega)$: DTFT of $y[n]$

$X(\omega)$: DTFT of $x[n]$

$G(\omega)$: Freq. response of DT LTI system

$$\boxed{\omega \equiv \Omega \cdot T}$$

So

$$Y_c(\Omega) = H_{LP}(\Omega)\, G(\Omega \cdot T)\, X(\Omega \cdot T)$$

$H_{LP}(\Omega)$: CTFTs

$G(\Omega \cdot T)\, X(\Omega \cdot T)$: DTFTs

Recall:

$$X(\omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} X_c\left(\frac{\omega - 2\pi k}{T}\right)$$

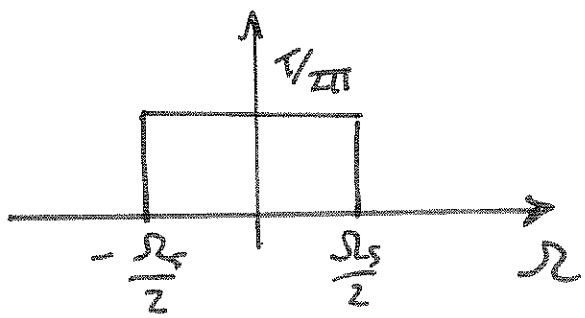
- or -

$$X(\Omega \cdot T) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} X_c\left(\Omega - k\Omega_s\right)$$

$\Rightarrow$

$$\boxed{Y_c(\Omega) = H_{LP}(\Omega) \cdot G(\Omega \cdot T) \cdot \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} X_c\left(\Omega - k\Omega_s\right)}$$

Now, if $X_c(\Omega)$ is bandlimited to $\left[-\frac{\Omega_s}{2}, \frac{\Omega_s}{2}\right]$ and we use the usual lowpass reconstruction filter in the D/A:



Then

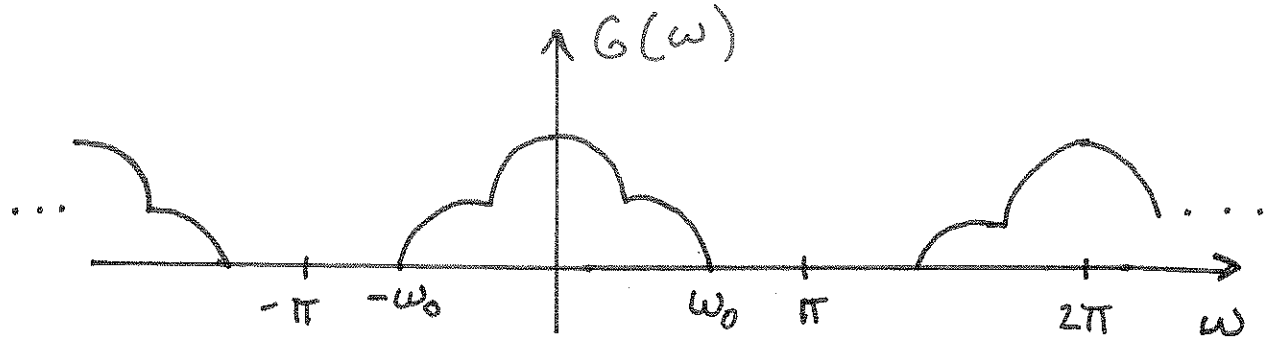
$$Y_c(\Omega) = \begin{cases} G(\Omega \cdot T)\, X_c(\Omega), & |\Omega| < \frac{\Omega_s}{2} \\ 0, & \text{otherwise} \end{cases}$$

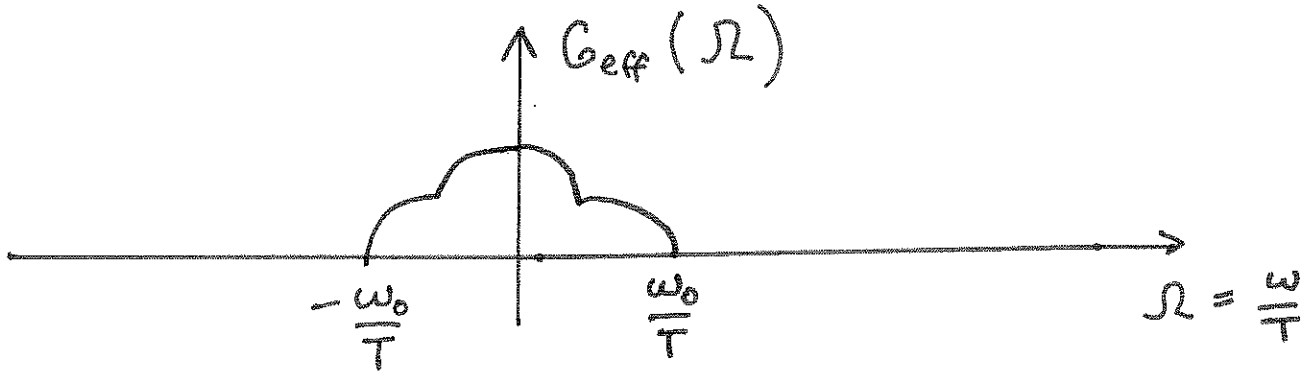
Summary: For bandlimited signals sampled at or above the Nyquist rate, we can relate the input and output of the DSP system by

$$Y_c(\Omega) = G_{eff}(\Omega)\, X_c(\Omega)$$

where

$$G_{eff}(\Omega) = \begin{cases} G(\Omega \cdot T), & |\Omega| < \frac{\Omega_s}{2} \\ 0, & \text{otherwise} \end{cases}$$



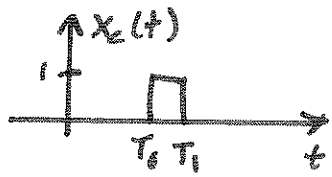


Note: $G_{eff}(\Omega)$ is LTI if and only if

1. $G(\omega)$ is LTI (in DT)

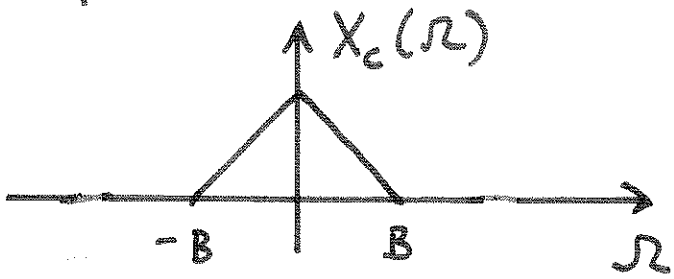
2. $x_c(t)$ is bandlimited and sampling rate is equal to or greater than Nyquist

Ex. $x_c(t) = u(t-T_0) - u(t-T_1)$, $T_1 > T_0$.



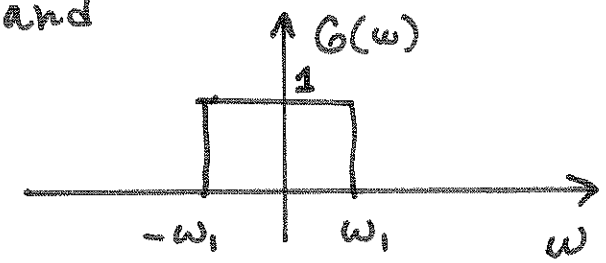
If the sampling period $T > T_1 - T_0$, then samples might "miss" the pulse... or they might not. $\Rightarrow$ time-varying behavior

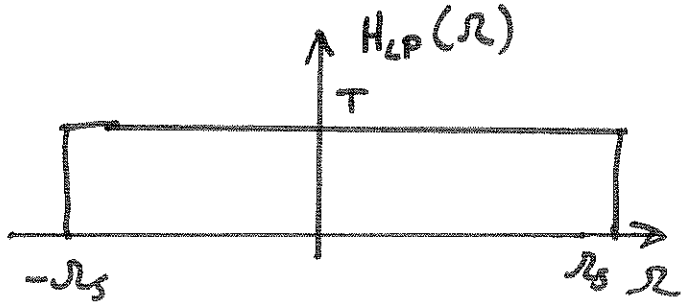
Ex. Suppose



$$\Omega_s = \frac{2\pi}{T}$$

and

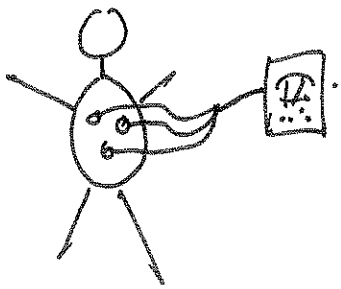


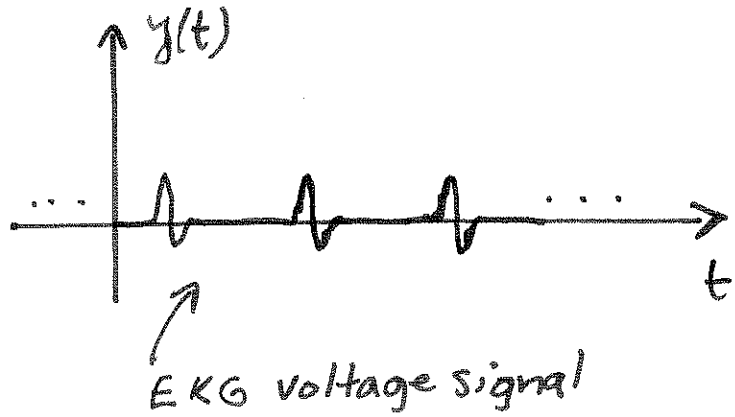


If $\frac{2\pi}{T} > 2B$ and $\omega_1 < BT$,

determine and sketch $Y_c(\Omega)$.

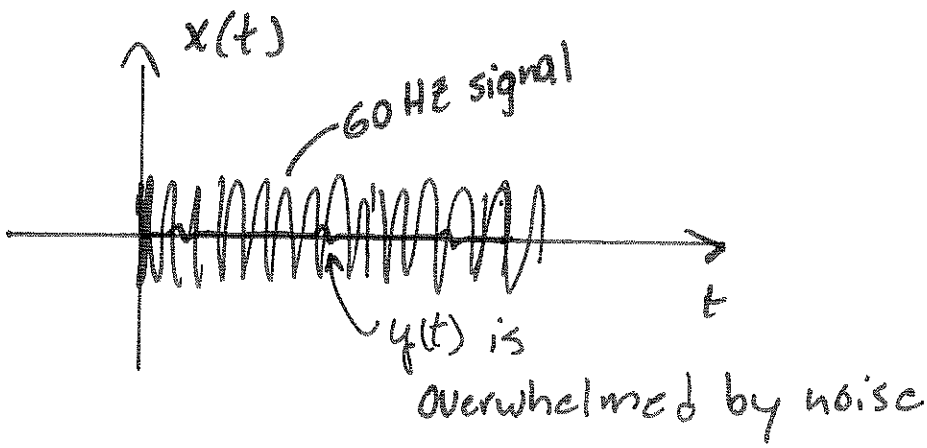
Application - 60 Hz Noise Removal



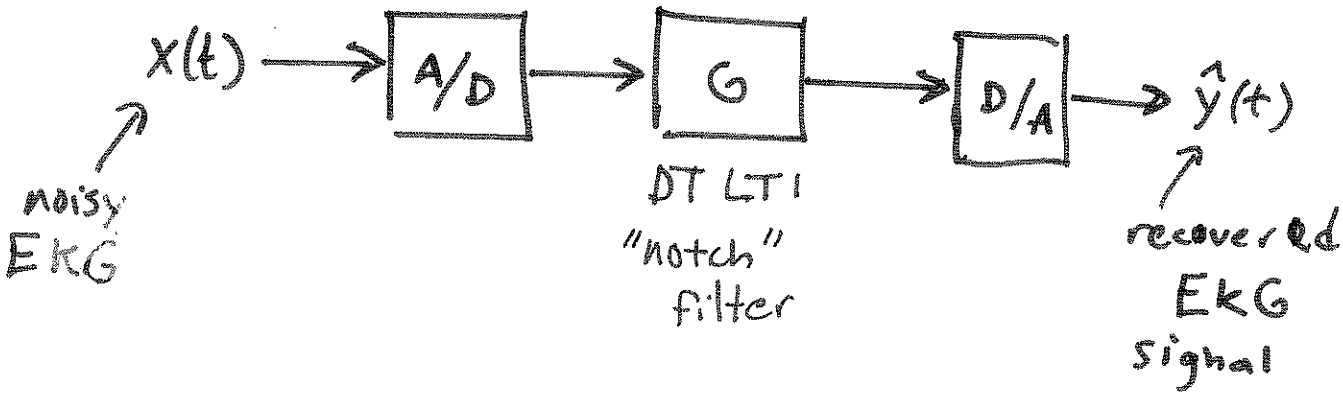


Unfortunately, in real-world situations electrodes also pick up ambient 60 Hz signals from lights, computers, etc.

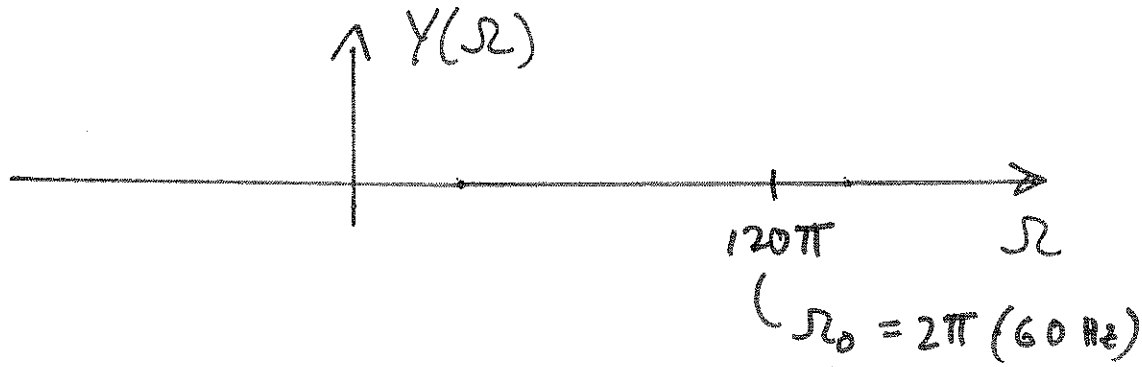
In fact, usually this "60 Hz noise" is much greater in amplitude than the EKG signal.



DSP Solution:



EKG Spectrum:



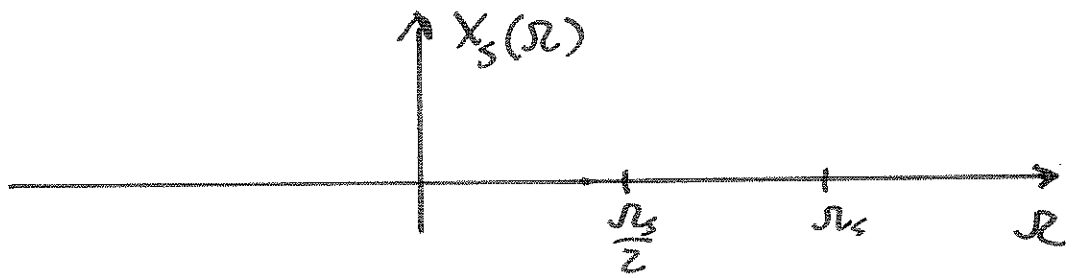
($\Omega_0 = 2\pi$ (60 Hz))

Line Noise Spectrum:

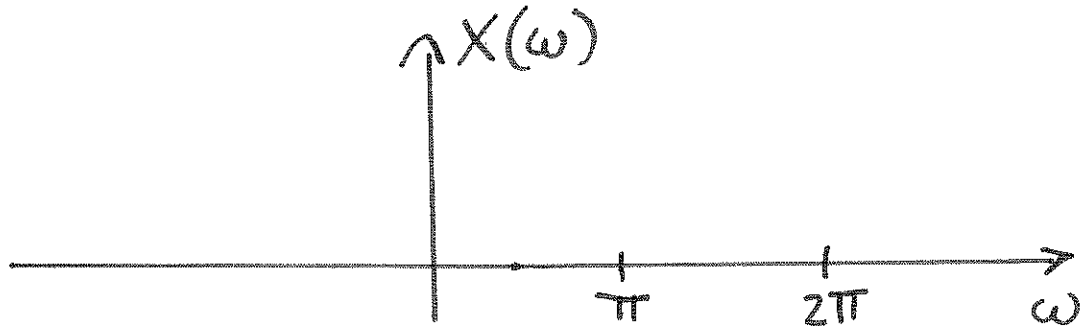
Sampling Period / Rate:

Sampled EKG:

$$x_s(t) \longleftrightarrow X_s(\Omega)$$

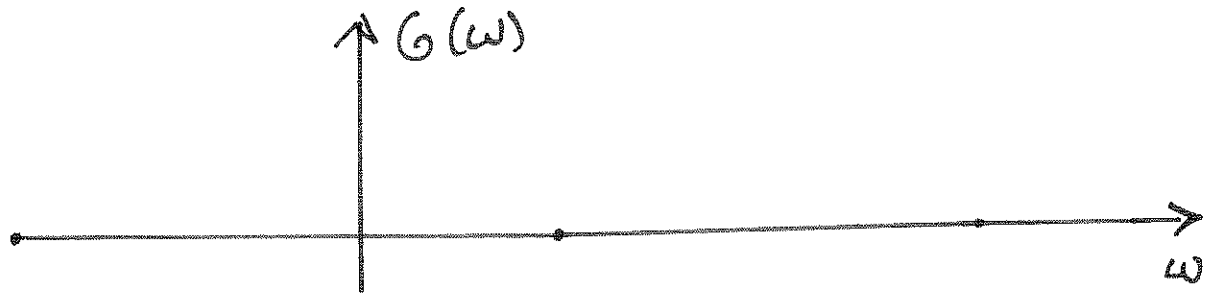


DT EKG:



Digital Filter:

Remove 60 Hz component —
preserve the rest



Discrete Fourier Transform (DFT)

Chapter 8 & 9 OSB

Outline

I. Discrete Fourier Transform

II. DFT Properties

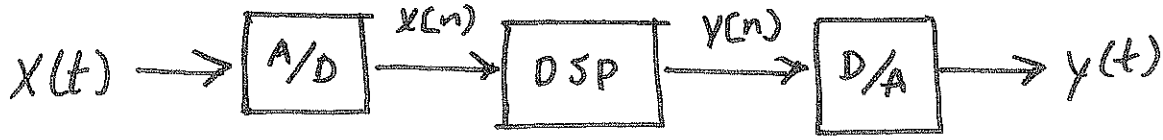
III. Circular Convolution

IV. Fast Fourier Transform (FFT) Algorithm

THEME

Transforms and algorithms for computer-based frequency domain analysis

We just covered ideal (and non-ideal) (time) sampling of CT signals. This enabled DT signal processing solutions for CT applications:



Much of the theoretical analysis of such systems relied on frequency domain representations.

How do we carry out frequency domain analysis on the computer?

Recall

$$x[n] \overset{\text{DTFT}}{\longleftrightarrow} X(\omega)$$ ← continuous freq variable

$$x(t) \overset{\text{CTFT}}{\longleftrightarrow} X(\Omega)$$ ← continuous freq. variable

Consider the DTFT of a DT signal $x[n]$. Assume $x[n]$ is of finite duration N (ie., an N-pt signal)

$$X(\omega) = \sum_{n=0}^{N-1} x[n] e^{-j\omega n}$$

$X(\omega)$: continuous function, indexed by real-valued parameter $-\pi \le \omega \le \pi$

$x[n]$: discrete function, indexed by integers

We want to work with $X(\omega)$ on a computer.

Why not <u>sample</u> $X(\omega)$?

Sample at $\omega = \frac{2\pi}{N} k$, $k = 0, 1, \ldots, N-1$

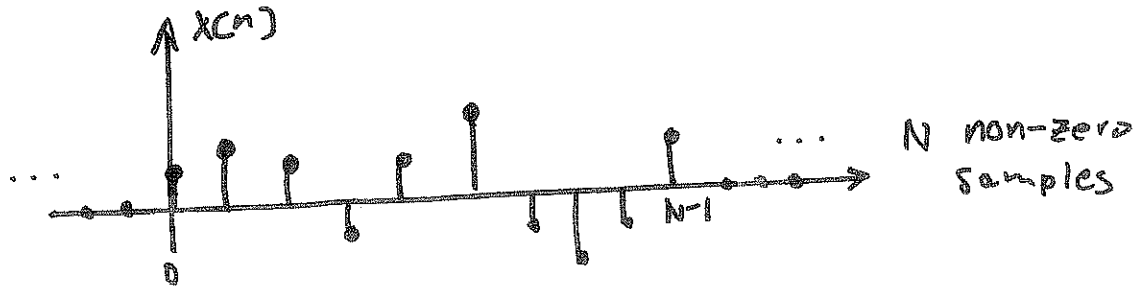
$$X_k = X\left(\frac{2\pi}{N} k\right)$$

$$= \sum_{n=0}^{N-1} x[n] e^{-j 2\pi \frac{k}{N} n}$$

$X[k]$, $k = 0, \ldots, N-1$, is called the <u>Discrete Fourier Transform</u> (DFT) of $x[n]$

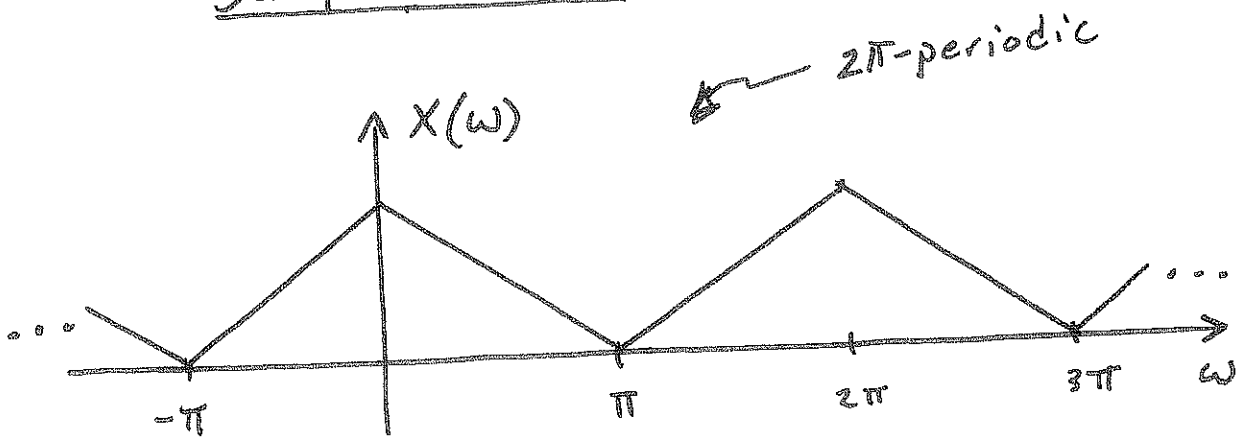
Ex.

Finite duration DT signal:



$$\text{DTFT} \quad X(\omega) = \sum_{n=0}^{N-1} x[n]\, e^{-j\omega n}, \quad -\pi \le \omega \le \pi \text{ —or— } 0 \le \omega \le 2\pi \text{ —or— any other } 2\pi\text{-interval}$$

Sample $X(\omega)$:



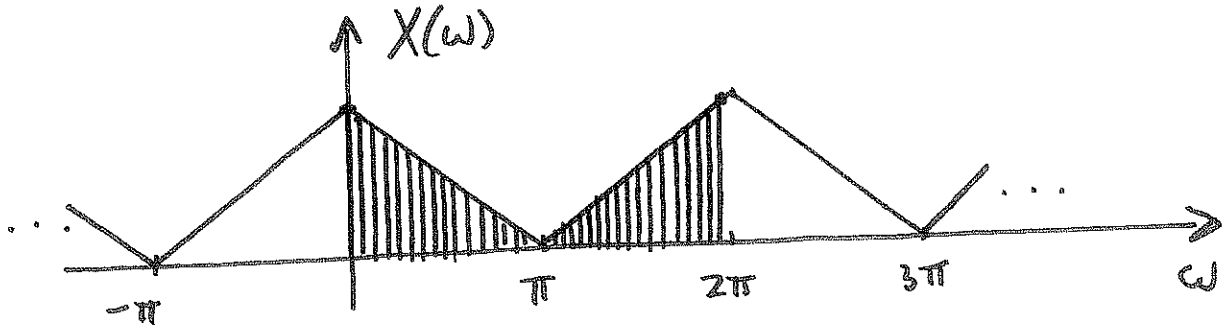
Sample at $\omega = \frac{2\pi k}{M}$, $k = 0, 1, \ldots, M-1$

For example, take $M = 10$.

(This is precisely how we would plot $X(\omega)$ in Matlab.)

Choosing M (the number of samples in 2π interval)

Case 1: Given N (length of $x[n]$), choose $M >> N$ to obtain a dense sampling of the DTFT



Case 2: Choose M as small as possible (to minimize the amount of computation)

In general, we require $M \geq N$ in order to represent all information in

$$x[n],\ n=0,\ldots,N-1$$

Let's concentrate on $M=N$:

$$x[n],\ n=0,\ldots,N-1 \overset{\text{DFT}}{\longleftrightarrow} X[k],\ k=0,\ldots,N-1$$

numbers $\longleftrightarrow$ N numbers

DFT

Define

$$X[k] \equiv X\left(\frac{2\pi k}{N}\right) \qquad k = 0, \ldots, N-1 \quad (M=N \text{ case})$$

($X[k]$: DFT; $X(\omega)$: DTFT)

where $N = \text{length}(x[n])$

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\left(2\pi \frac{k}{N}\right)n} \qquad \text{DFT}$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]\, e^{+j\left(2\pi \frac{k}{N}\right)n} \qquad \text{Inverse DFT (IDFT)}$$

Interpretation: Represent $x[n]$ in terms of a sum of N complex sinusoids of amplitudes $X[k]$ and frequencies

$$\omega_k = \frac{2\pi k}{N}, \quad k = 0, \ldots, N-1$$

(Think: Fourier series with fundamental freq $\frac{2\pi}{N}$)

Remarks:

① IDFT treats $x[n]$ as though it were N-periodic.

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j \frac{2\pi k}{N} n}$$

$n = 0, \ldots, N-1$

What about other values of n?

$x[n+N] =$

(2) Proof that the IDFT inverts the DFT for $n = 0, \ldots, N-1$

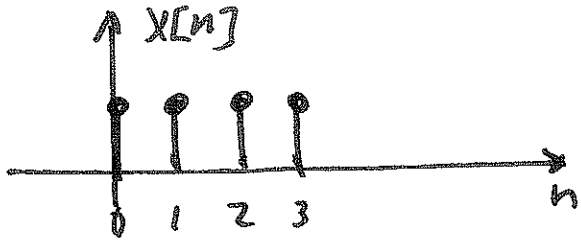
$$\frac{1}{N} \sum_{k=0}^{N-1} X[k] \, e^{j \frac{2\pi k}{N} n}$$

$$X[k] = \sum_{m=0}^{N-1} x[m] \, e^{-j \frac{2\pi k}{N} m}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} \left(\sum_{m=0}^{N-1} x[m] \, e^{-j \frac{2\pi k}{N} m} \right) e^{j \frac{2\pi k}{N} n}$$

$$=$$

Ex. Compute DFT of



$N = 4$

Method 1: DFT Formula

$$X[k] = \sum_{n=0}^{N-1} x(n)\, e^{-j\frac{2\pi k}{N} n}$$

$$= 1 + e^{-j\frac{2\pi k}{4}} + e^{-j\frac{2\pi k}{4}\cdot 2} + e^{-j\frac{2\pi k}{4}\cdot 3}$$

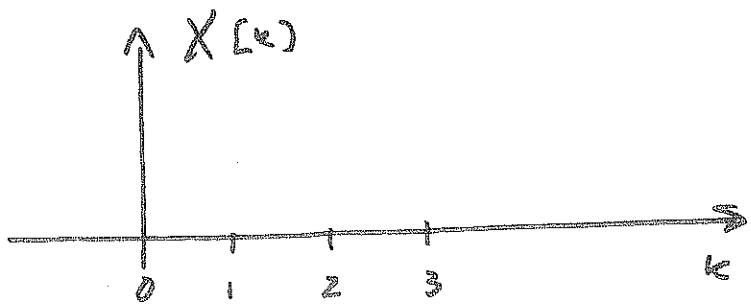
$$= 1 + e^{-j\frac{\pi}{2}k} + e^{-j\pi k} + e^{-j\frac{3}{2}\pi k}$$

$X[0] =$

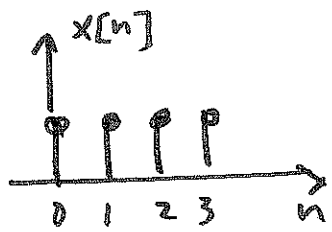
$X[1] =$

$X[2] =$

$X[3] =$



Method 2: Sample DTFT

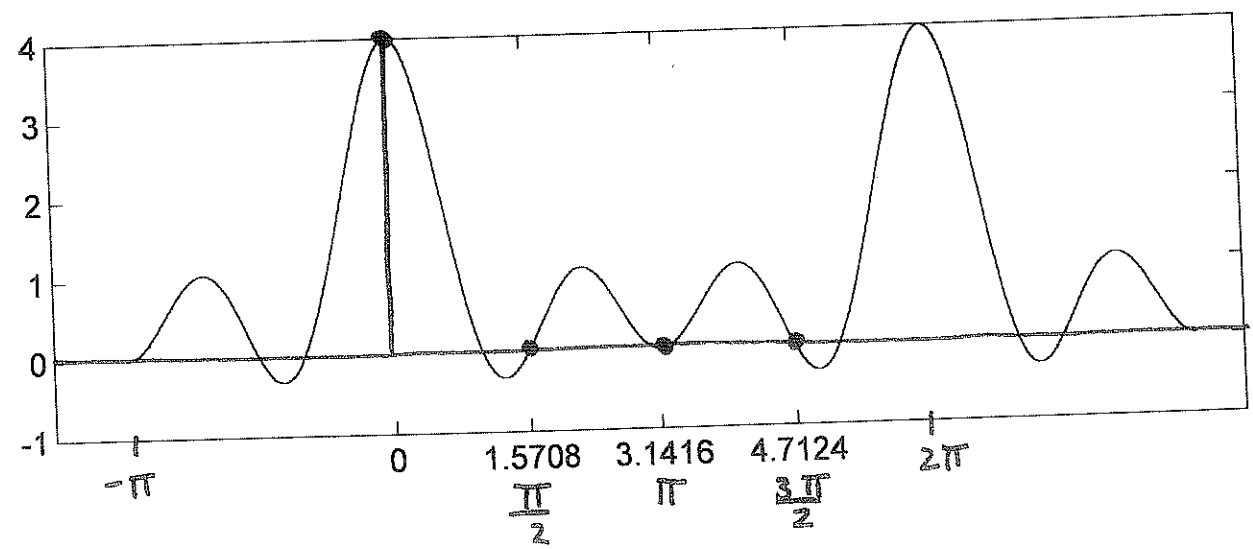


$\xleftrightarrow{\text{DTFT}}$

$$X(\omega) = \sum_{n=0}^{3} e^{-j\omega n}$$

$$= \frac{1 - e^{-j4\omega}}{1 - e^{-j\omega}}$$

$$=$$



Sample points:

$$\omega_k = \frac{2\pi k}{4} = \frac{\pi}{2} k \quad , \quad k = 0, 1, 2, 3$$

Periodicity of the DFT

DFT $X[k]$ consists of <u>samples</u> of DTFT

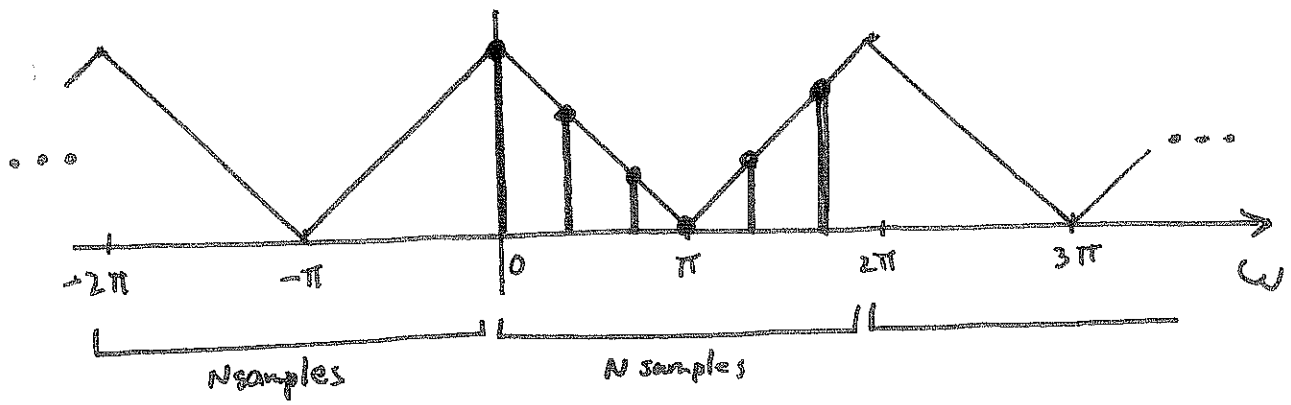
So,

$X(\omega)$ 2π-periodic $\Longrightarrow$ $X[k]$ N-periodic

DTFT DFT

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi k}{N}n}$$

↖ N-periodic basis functions



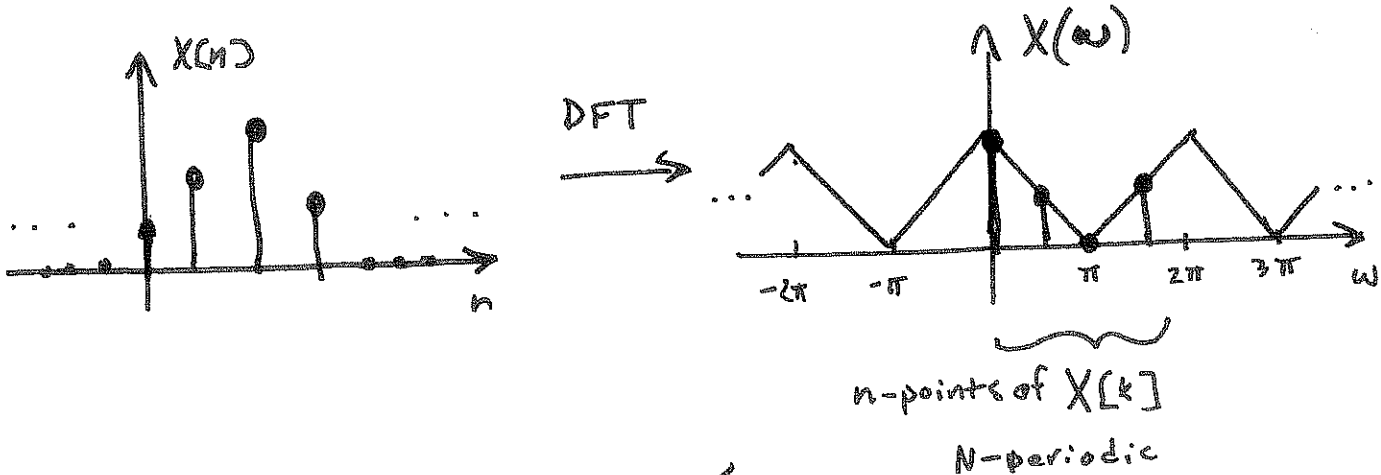
Also recall,

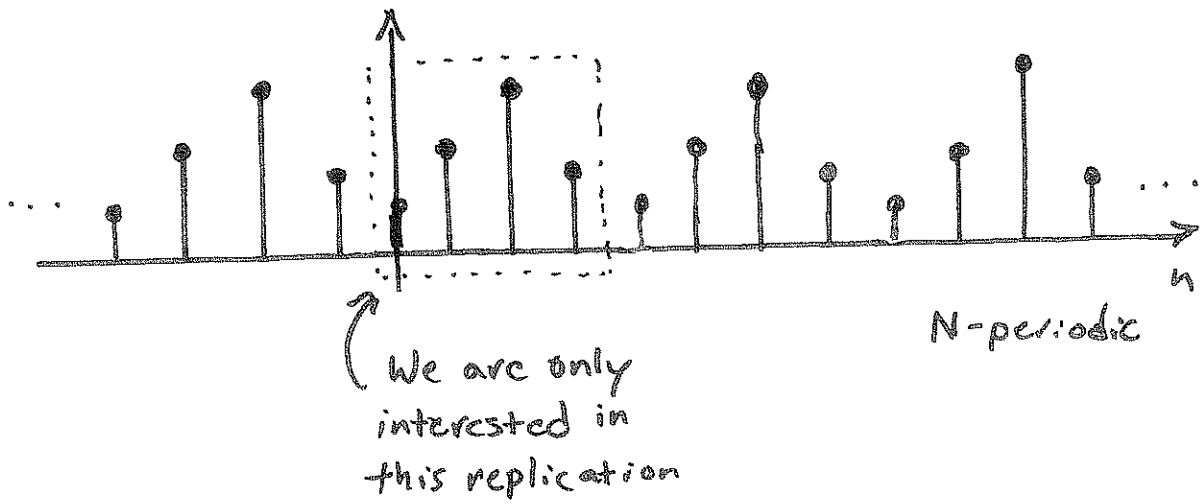
$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\, e^{j\frac{2\pi k}{N}n}$$

$$= \frac{1}{N}\sum_{k=0}^{N-1} X[k]\, e^{j\frac{2\pi k}{N}(n+mN)}$$

$$=$$

Illustration:

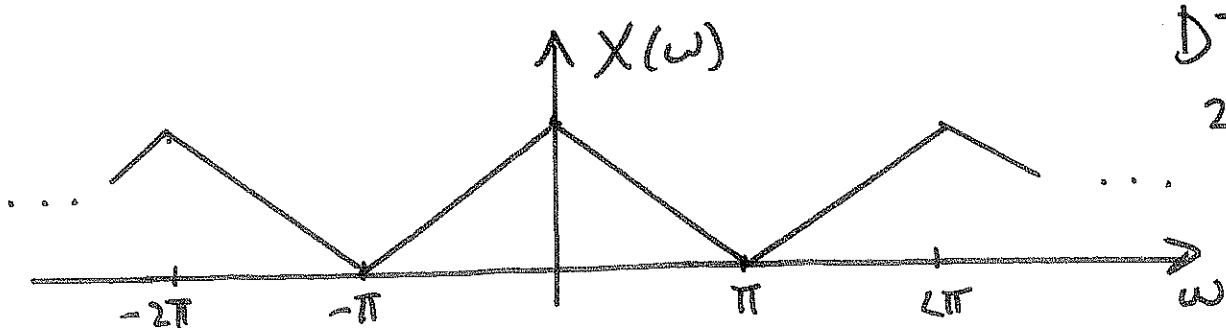


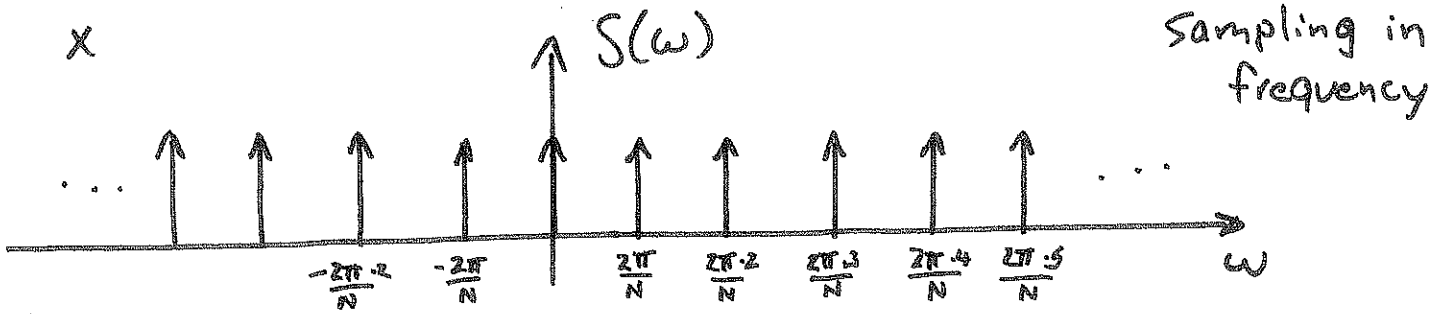


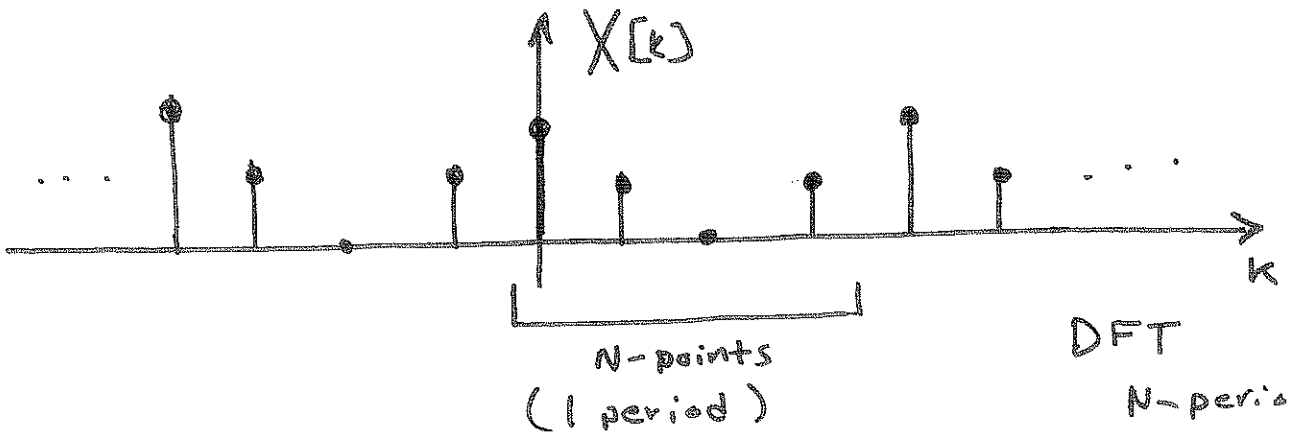
$\Rightarrow$ When we deal with DFT, we need to remember that, in effect, this treats the signal as an N-periodic sequence.

A Sampling Perspective

Think of sampling the continuous function $X(\omega)$.







Recall: Mult in freq $\Longleftrightarrow$ Convolution in time

Inverse DTFT of $S(\omega)$:

$$\sum_{k=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi k}{N}\right) \overset{\text{DTFT}}{\longleftrightarrow} N \sum_{m=-\infty}^{\infty} \delta[n - mN] \equiv S[n]$$

↑ Why?

$S[n]$ is N-periodic, so it has a Fourier series:

$$C_k = \frac{1}{N} \int_{-\frac{N}{2}}^{\frac{N}{2}} \delta[n]\, e^{-j\frac{2\pi k}{N} \cdot n}$$

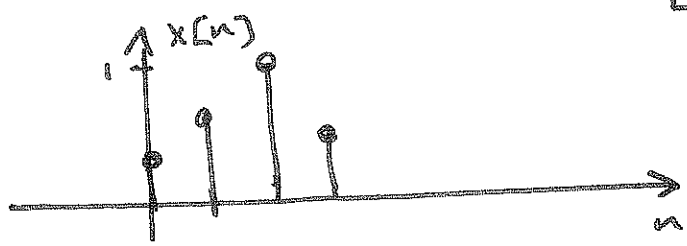
$$= \frac{1}{N}$$

$$\Rightarrow S[n] = \sum_{k=-\infty}^{\infty} e^{-j\frac{2\pi k}{N} \cdot n}$$

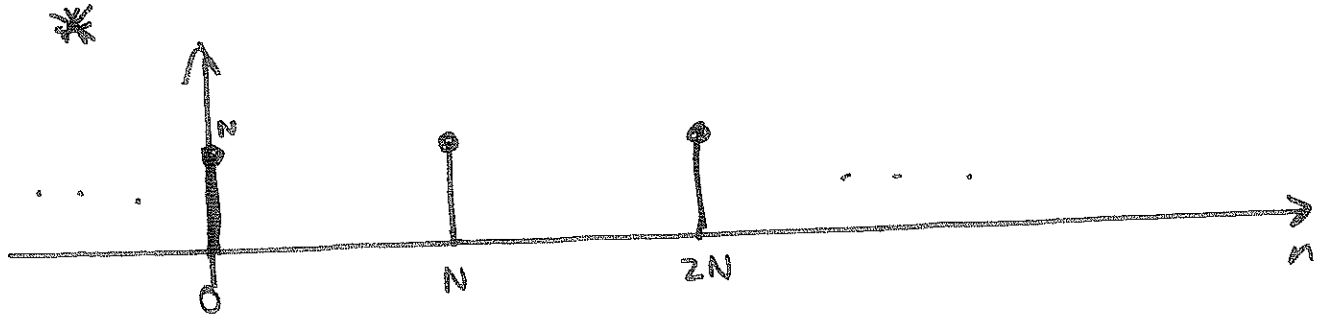
$\updownarrow$ DTFT

$\delta\left(\omega - \frac{2\pi k}{N}\right)$

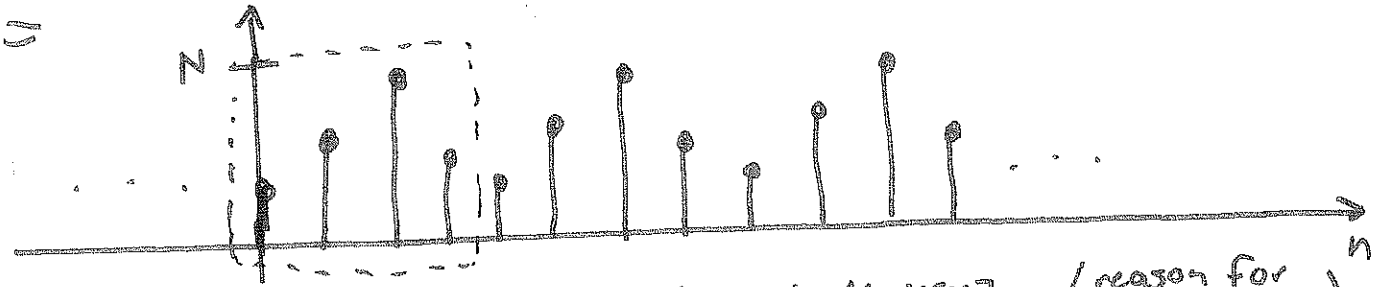
So, in the time-domain we have ↓



$*$



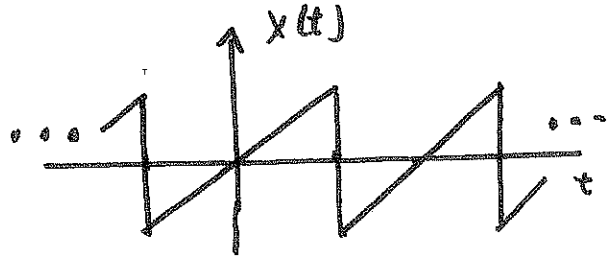
$=$

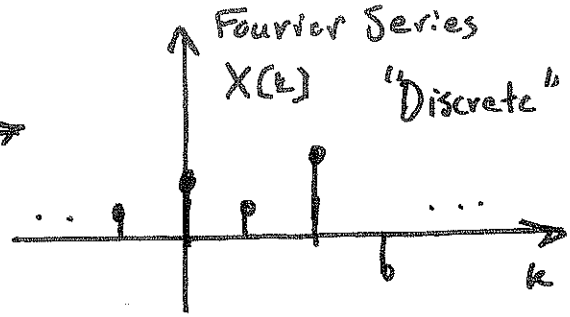


↑ this piece is just $N \cdot x[n]$ (reason for $\frac{1}{N}$ in IDFT)

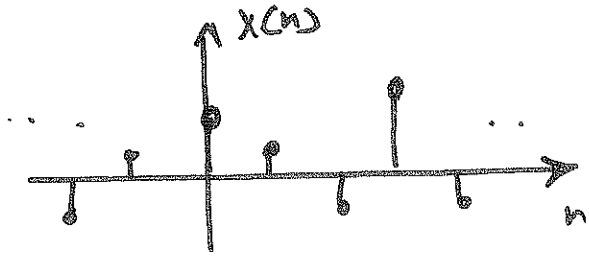
Connections

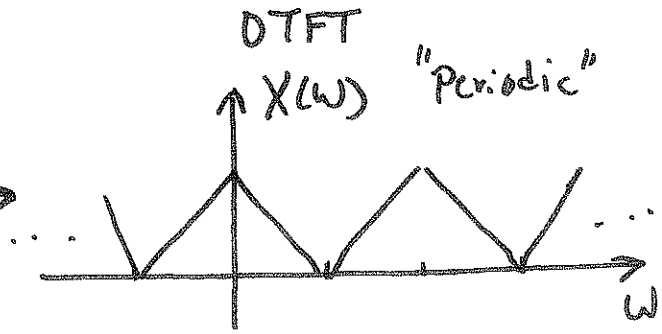
Periodic CT signal



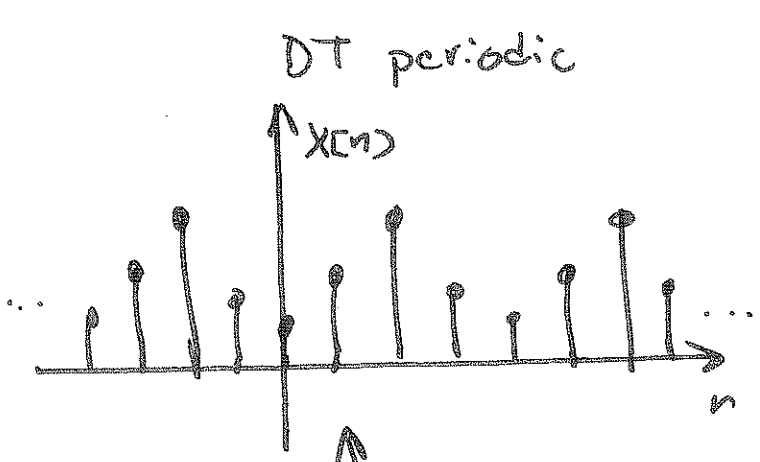


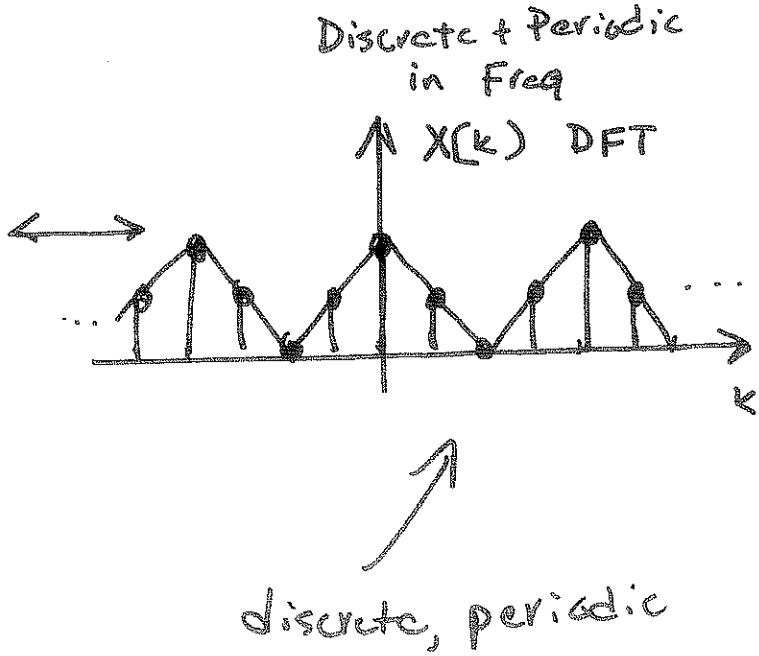
DT Signal





Combine to see...





DFT PROPERTIES

Linearity

$$A\,x_1[n] + B\,x_2[n] \overset{DFT}{\longleftrightarrow} A\,X_1[k] + B\,X_2[k]$$

Shift

$$x[n-m] \overset{DFT}{\longleftrightarrow} X[k]\,e^{-j\,2\pi\left(\frac{k}{N}\right)m}$$

Modulation

$$x[n]\,e^{j2\pi\left(\frac{m}{N}\right)n} \overset{DFT}{\longleftrightarrow} X[k-m]$$

Symmetry (conjugate Symmetry)

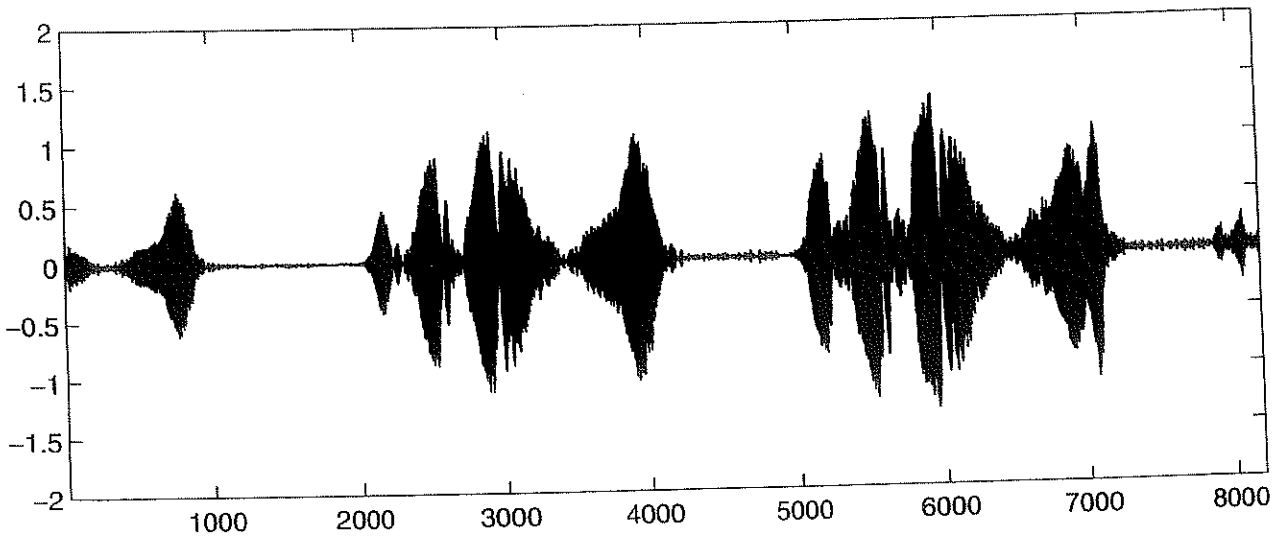
$$\text{Real } x[n] \Longrightarrow X^*[k] = X[N-k]$$

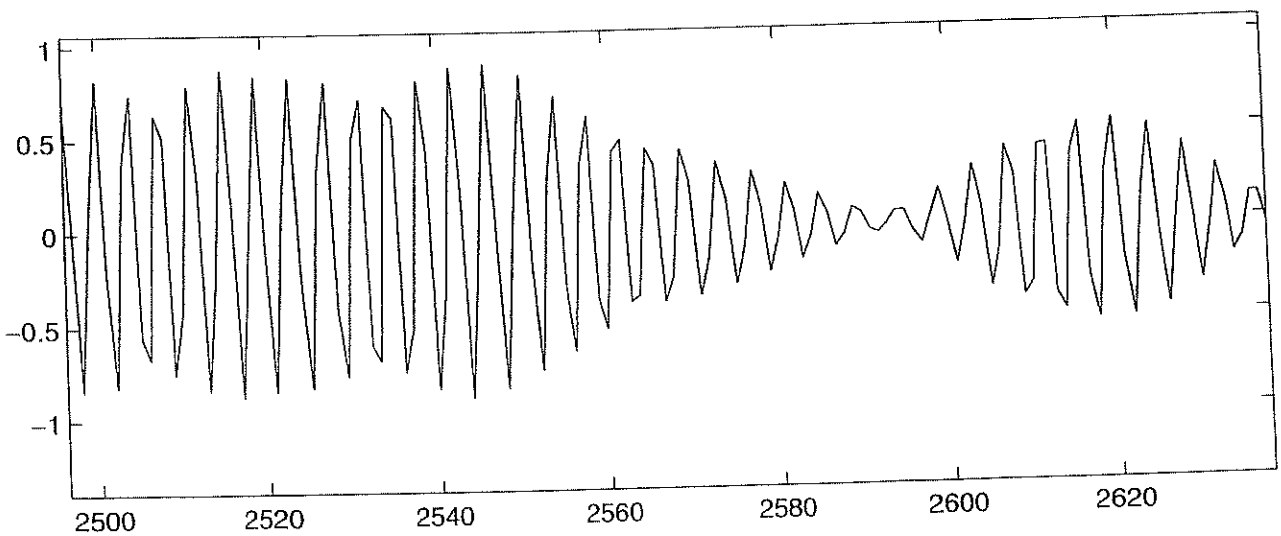
DFT PROPERTIES

Finite-Length Sequence (Length N)	N-Point DFT (Length N)
1. $x[n]$	$X[k]$
2. $x_1[n]$, $x_2[n]$	$X_1[k]$, $X_2[k]$
3. $ax_1[n] + bx_2[n]$	$aX_1[k] + bX_2[k]$
4. $X[n]$	$Nx[((-k))_N]$
5. $x[((n-m))_N]$	$W_N^{km}X[k]$
6. $W_N^{-\ell n}x[n]$	$X[((k-\ell))_N]$
7. $\sum_{m=0}^{N-1} x_1(m)x_2[((n-m))_N]$	$X_1[k]X_2[k]$
8. $x_1[n]x_2[n]$	$\frac{1}{N}\sum_{\ell=0}^{N-1} X_1(\ell)X_2[((k-\ell))_N]$
9. $x^*[n]$	$X^*[((-k))_N]$
10. $x^*[((-n))_N]$	$X^*[k]$
11. $\mathcal{R}e\{x[n]\}$	$X_{ep}[k] = \frac{1}{2}\{X[((k))_N] + X^*[((-k))_N]\}$
12. $j\mathcal{I}m\{x[n]\}$	$X_{op}[k] = \frac{1}{2}\{X[((k))_N] - X^*[((-k))_N]\}$
13. $x_{ep}[n] = \frac{1}{2}\{x[n] + x^*[((-n))_N]\}$	$\mathcal{R}e\{X[k]\}$
14. $x_{op}[n] = \frac{1}{2}\{x[n] - x^*[((-n))_N]\}$	$j\mathcal{I}m\{X[k]\}$
Properties 15–17 apply only when $x[n]$ is real.	
15. Symmetry properties	$\begin{cases} X[k] = X^*[((-k))_N] \\ \mathcal{R}e\{X[k]\} = \mathcal{R}e\{X[((-k))_N]\} \\ \mathcal{I}m\{X[k]\} = -\mathcal{I}m\{X[((-k))_N]\} \\ \lvert X[k]\rvert = \lvert X[((-k))_N]\rvert \\ \measuredangle\{X[k]\} = -\measuredangle\{X[((-k))_N]\} \end{cases}$
16. $x_{ep}[n] = \frac{1}{2}\{x[n] + x[((-n))_N]\}$	$\mathcal{R}e\{X[k]\}$
17. $x_{op}[n] = \frac{1}{2}\{x[n] - x[((-n))_N]\}$	$j\mathcal{I}m\{X[k]\}$

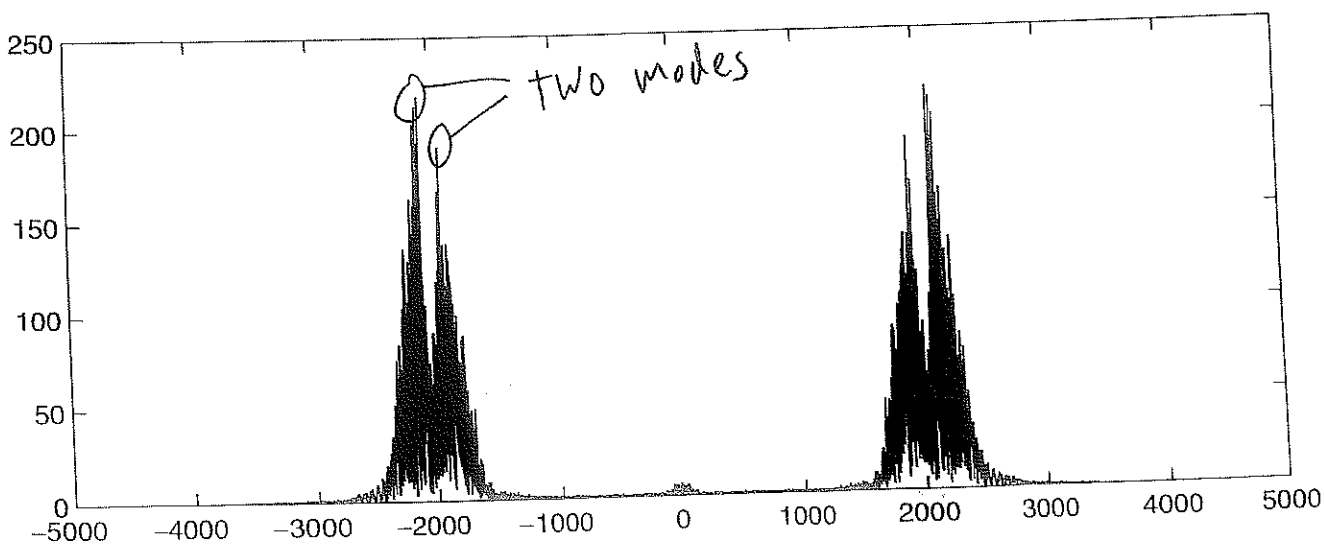
Bird song

ime series

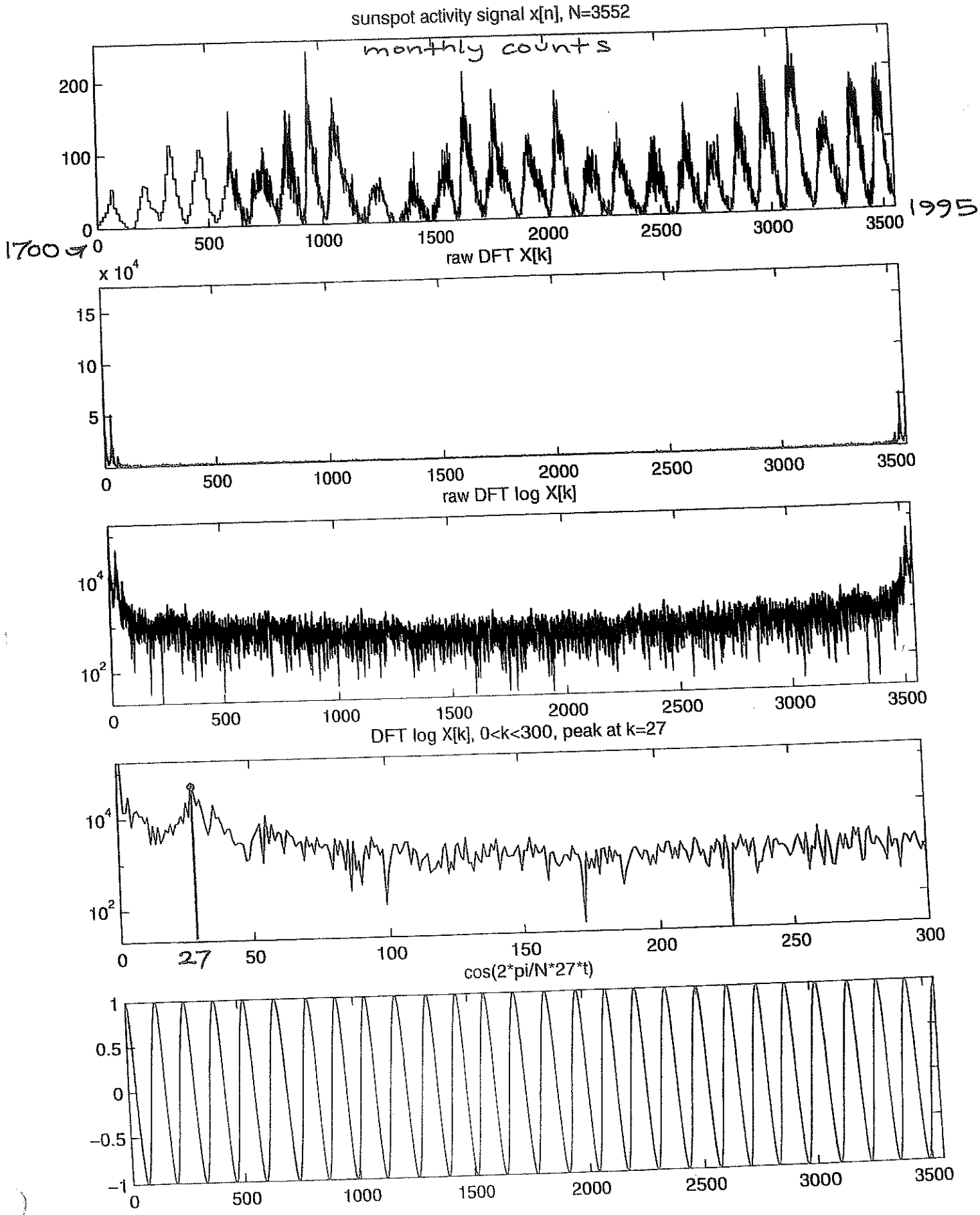




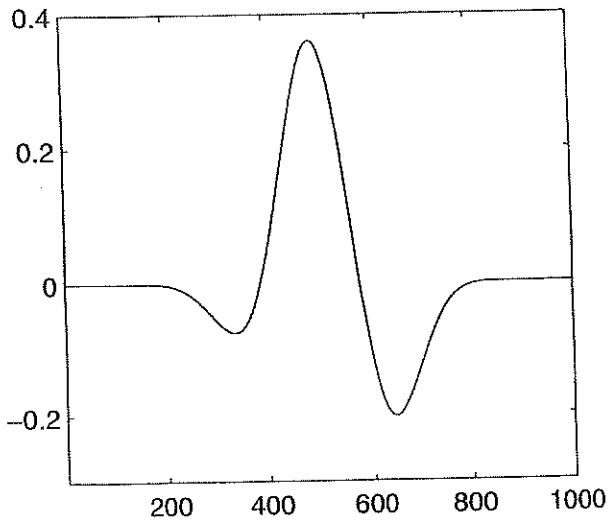
DFT



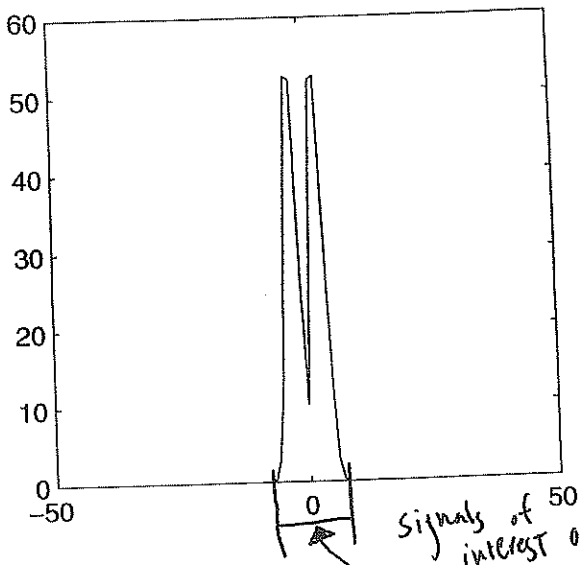
DFT SPECTRAL ANALYSIS



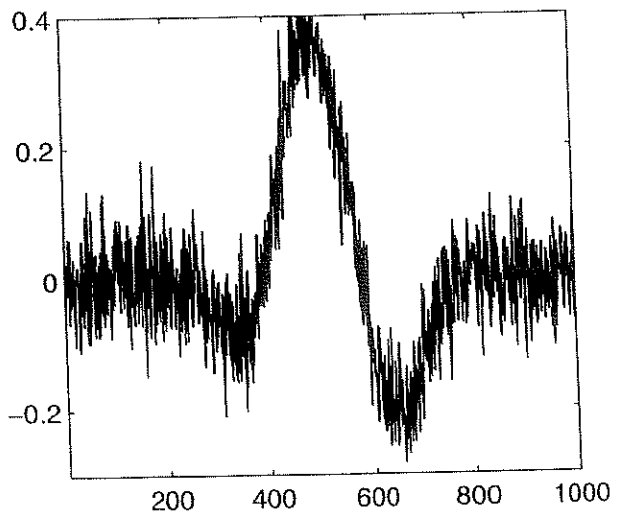
Signal recovery



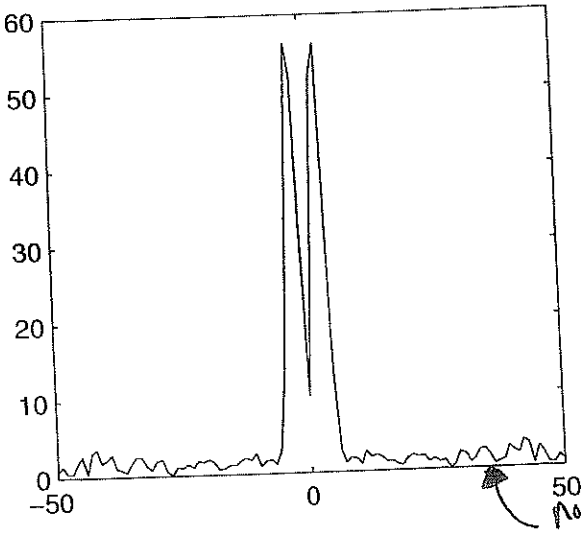
original signal



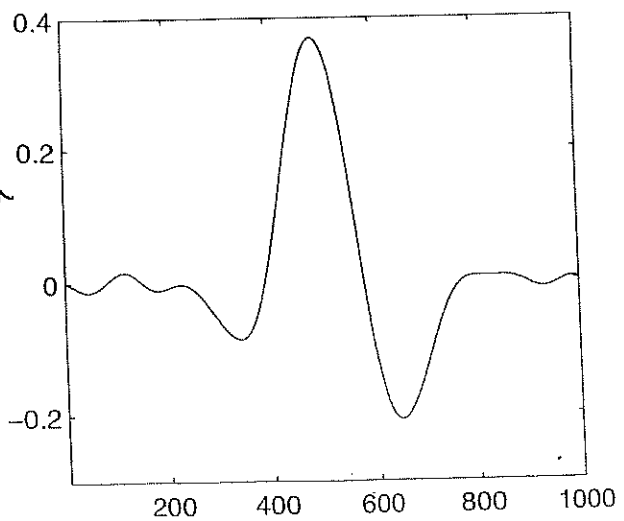
Signals of interest occupy this freq. band



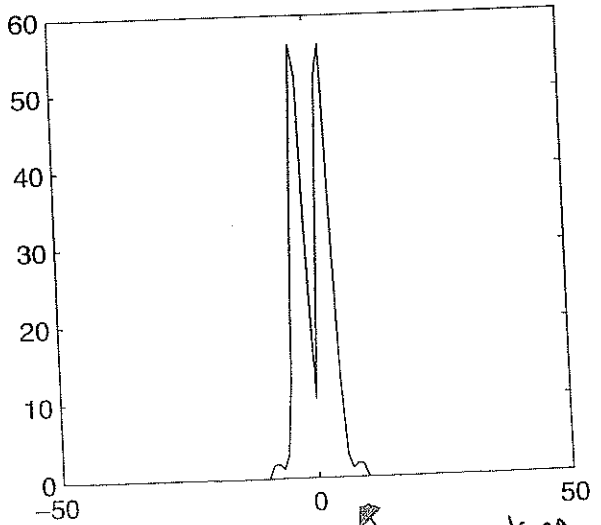
noisy signal



noise is spread out in freq.



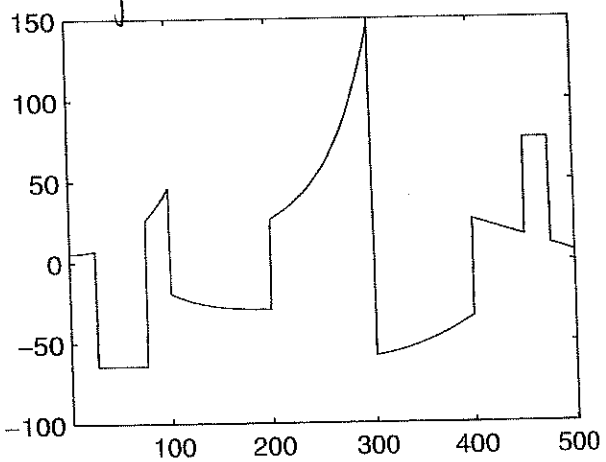
recovered signal



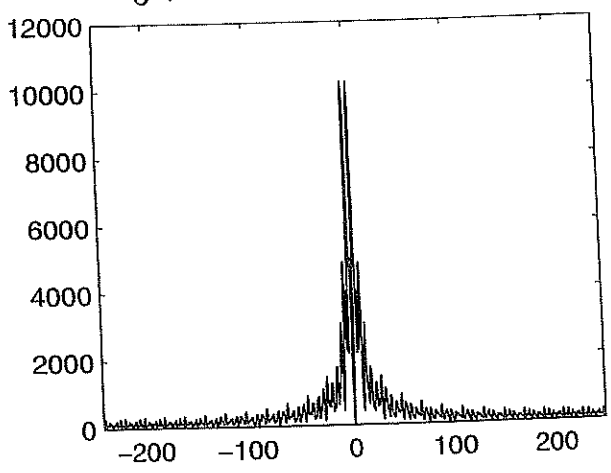
just keep the frequencies we're interested in

Compression (1-D)

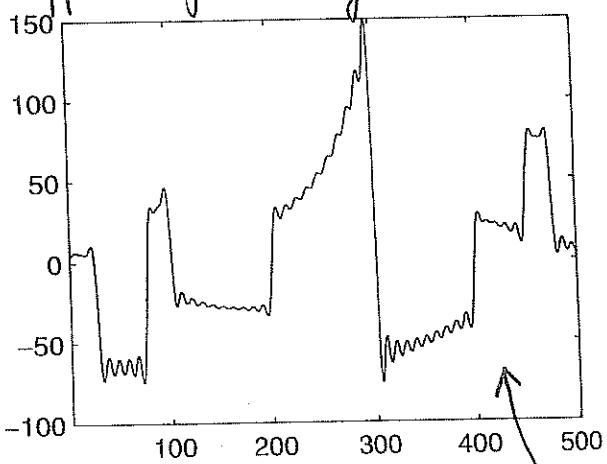
Signal 500 points

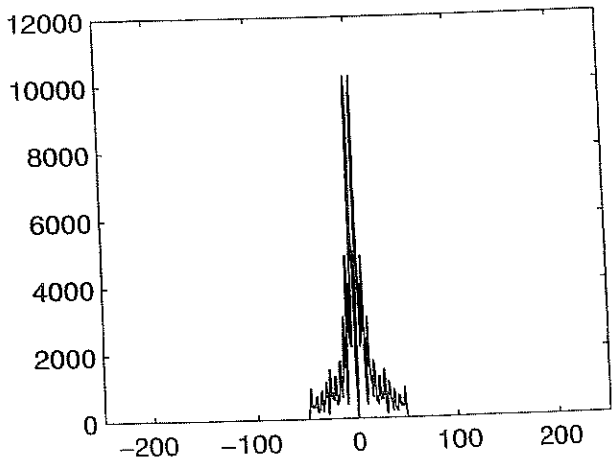


DFT



approx. signal using 100 DFT coeffs.





compressed by 80%
(100 numbers v. 500)

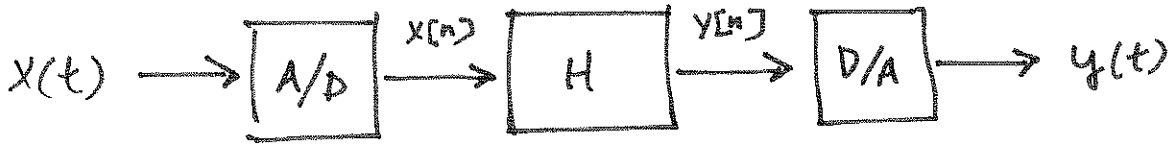
Image compression

256×256 image (65,536 pixels)

reconstructed using only 6000 Fourier coefficients →

We've cut down on storage space by >90%
(while incurring some loss)

Filtering with the DFT



$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k]$$

$$Y(\omega) = X(\omega) \cdot H(\omega)$$

Assume that $H(\omega)$ is specified.

How can we implement $X(\omega) \cdot H(\omega)$ in a computer?

Answer: Discretize (sample) $X(\omega)$ and $H(\omega)$.

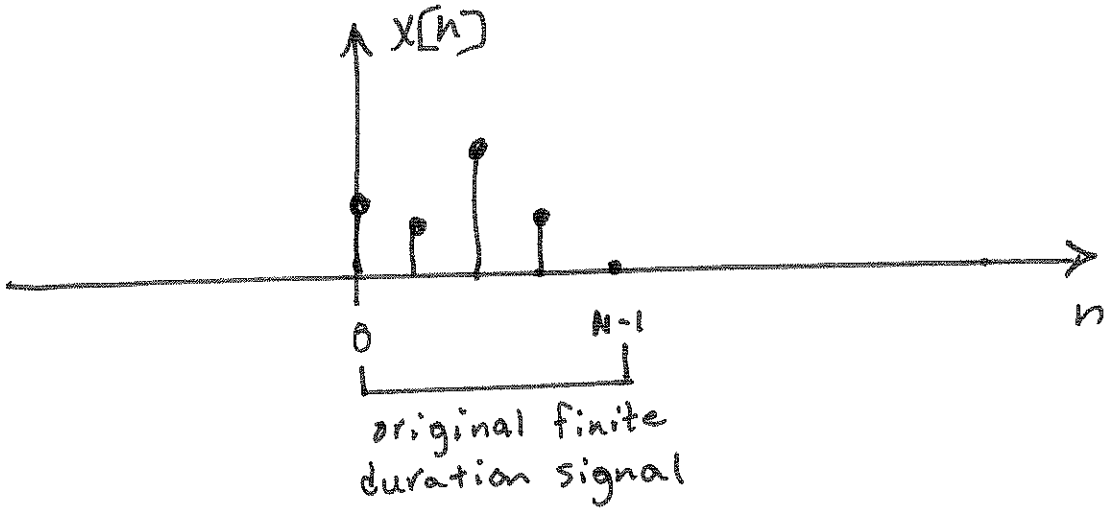
$\Rightarrow$ Take DFTs of $x[n]$ and $h[n]$ to get $X[k]$ and $H[k]$.

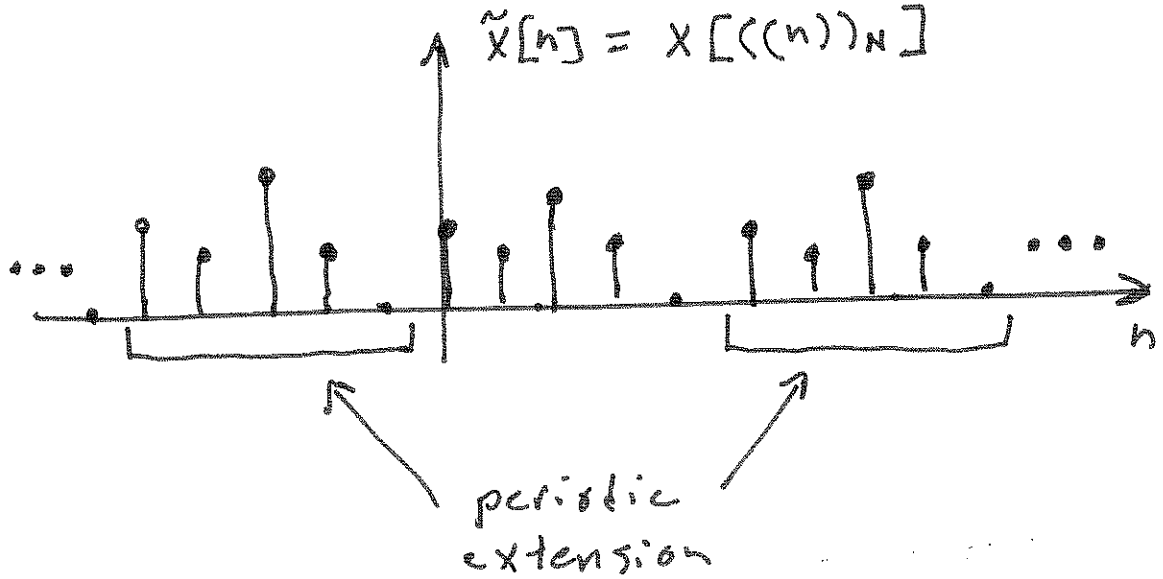
Then compute

$$\tilde{y}[n] = \text{IDFT}\left(X[k] \cdot H[k]\right)$$

Does $\tilde{y}[n] = y[n]$?

Recall that the DFT treats N-point sequences as if they are periodically extended:





Compute IDFT of $Y[k] = H[k] \cdot X[k]$

$$\tilde{y}[n] = \frac{1}{N} \sum_{k=0}^{N-1} Y[k] e^{j 2\pi \left(\frac{k}{N}\right) n}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} X[k] H[k] e^{j 2\pi \left(\frac{k}{N}\right) n}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} \left(\sum_{m=0}^{N-1} x[m] e^{-j 2\pi \left(\frac{k}{N}\right) m} \right) H[k] e^{j 2\pi \left(\frac{k}{N}\right) n}$$

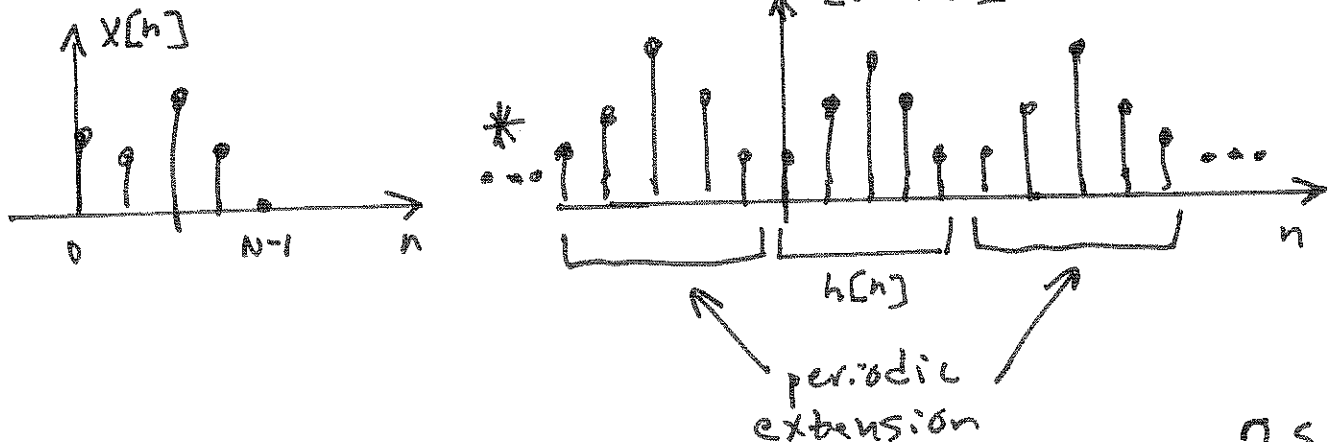
$$= \sum_{m=0}^{N-1} x[m] \left(\frac{1}{N} \sum_{k=0}^{N-1} H[k] e^{j 2\pi \left(\frac{k}{N}\right)(n-m)} \right)$$

$$= \sum_{m=0}^{N-1} x[m] \, h[((n-m))_N]$$

$\hat{h}[n-m] = h[((n-m))_N]$

↳ IDFT periodically extends $h[n]$

This computes



$$\tilde{y}[n] = \sum_{m=0}^{N-1} x[m]\, h[((n-m))_N]$$

is called <u>circular convolution</u>

and is denoted by

$$\tilde{y}[n] = x[n] \textcircled{N} h[n]$$

↑ for N-periodic extension

DFT Pair

$$x[n] \textcircled{N} h[n] \overset{\text{DFT}}{\longleftrightarrow} X[k] \cdot H[k]$$

Note:

In general

$$x[n] \textcircled{N} h[n] \neq x[n] * h[n]$$

Ex. $x[n] = \{1, 2, 3\}$ length 3 signals

$h[n] = \{1, 0, 2\}$

$x[n] = \{\ldots, 0, 1, 2, 3, 0, \ldots\}$

$h[n] = \{\ldots, 0, 1, 0, 2, 0, \ldots\}$

Regular Convolution:

$$y[n] = \sum_{m=0}^{2} x[m]\, h[n-m]$$

$n = 0$: $\cdots 0, 0, 0, \underset{\times}{1}, 2, 3, 0 \cdots$

$\cdots 0, 2, 0, 1, 0, 0, 0, \cdots$

$y[0] = 1$

$n = 1$: $\cdots 0, 0, \underset{\times}{1}, 2, 3, 0 \cdots$

$\cdots 0, 2, 0, 1, 0, 0 \cdots$

$y[1] = 2$

$y[2] =$

$y[3] =$

$y[4] =$

Circular Convolution:

$$\tilde{y}[n] = \sum_{m=0}^{2} x[m]\, h[((n-m))_N]$$

$$h[((n))_N] = \{\ldots, 1, 0, 2, 1, 0, 2, 1, 0, 2, \ldots\}$$

$n=0$:

$\ldots, 0, 0, 0, 1, 2, 3, 0, \ldots$

$\ldots, 1, 2, 0, 1, 2, 0, 1, \ldots$

$$\tilde{y}[0] = 5$$

$n=1$:

$\ldots, 0, 0, 0, 1, 2, 3, 0, \ldots$

$\ldots, 0, 1, 2, 0, 1, 2, 0, 1, \ldots$

$$\tilde{y}[1] = 8$$

$n=2$:

$\ldots, 0, 0, 0, 1, 2, 3, 0, \ldots$

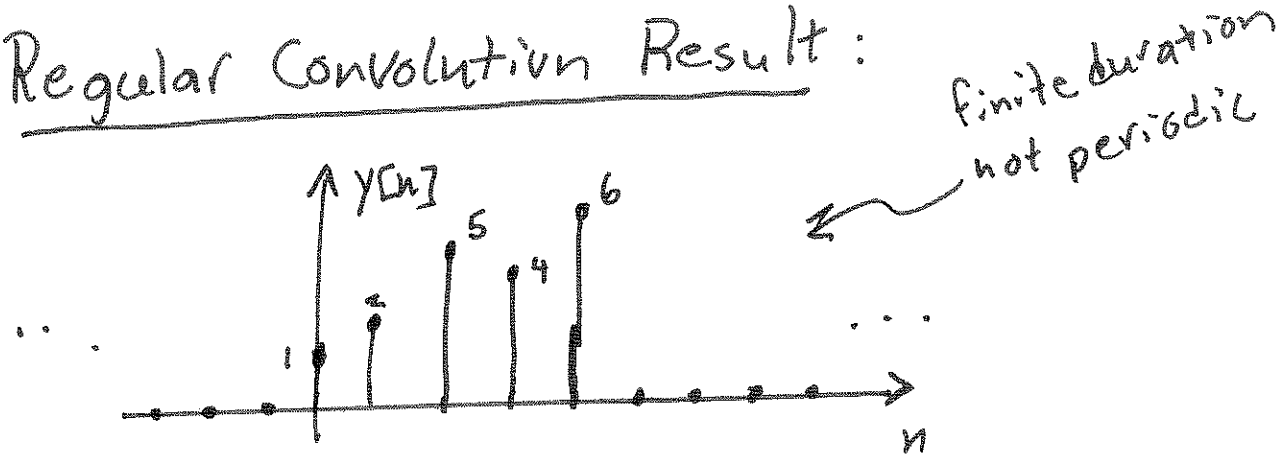
$\ldots, 0, 0, 1, 2, 0, 1, 2, 0, \ldots$

$$\tilde{y}[2] = 5$$

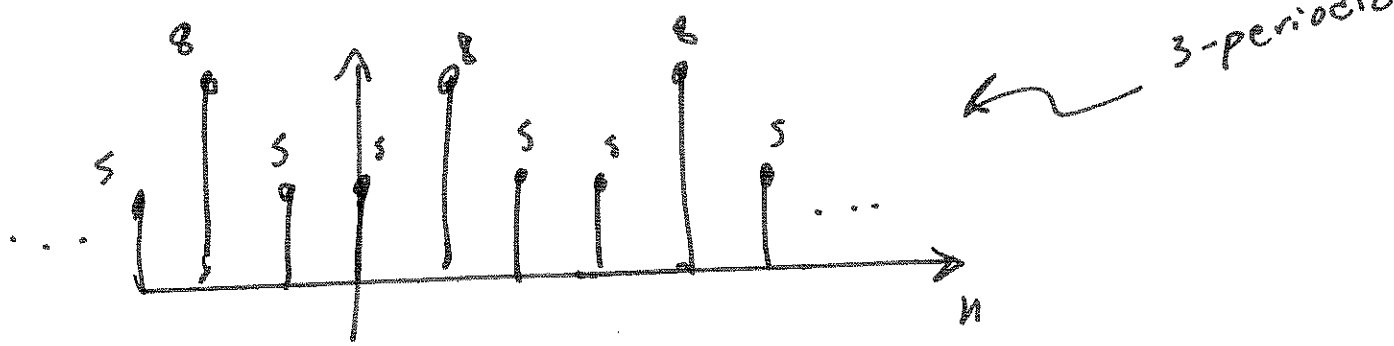
$\tilde{y}[3] =$

$\tilde{y}[4] =$

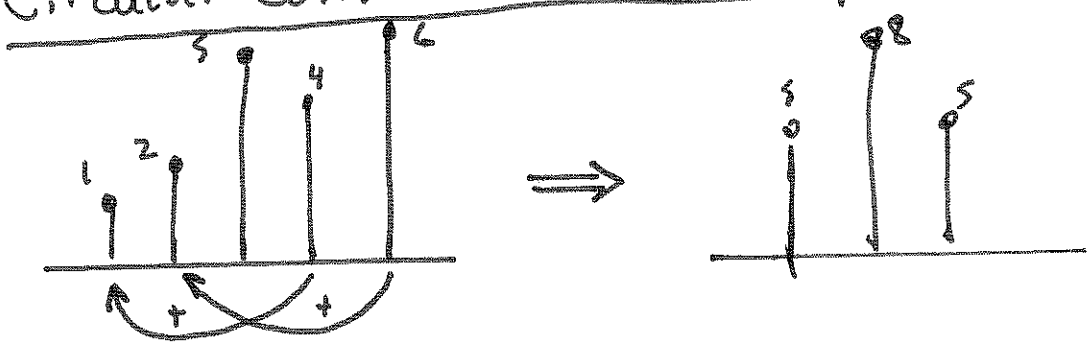
Regular Convolution Result:



Circular Convolution Result:



Circular Convolution from Regular:



"wrap-around" effect due to periodic extension

Regular Convolution from Periodic Convolution:

"zero-pad" $x[n]$ and $h[n]$ to avoid overlap / wrap-around effect:

$$x[n] = \{1, 2, 3, 0, 0\}$$

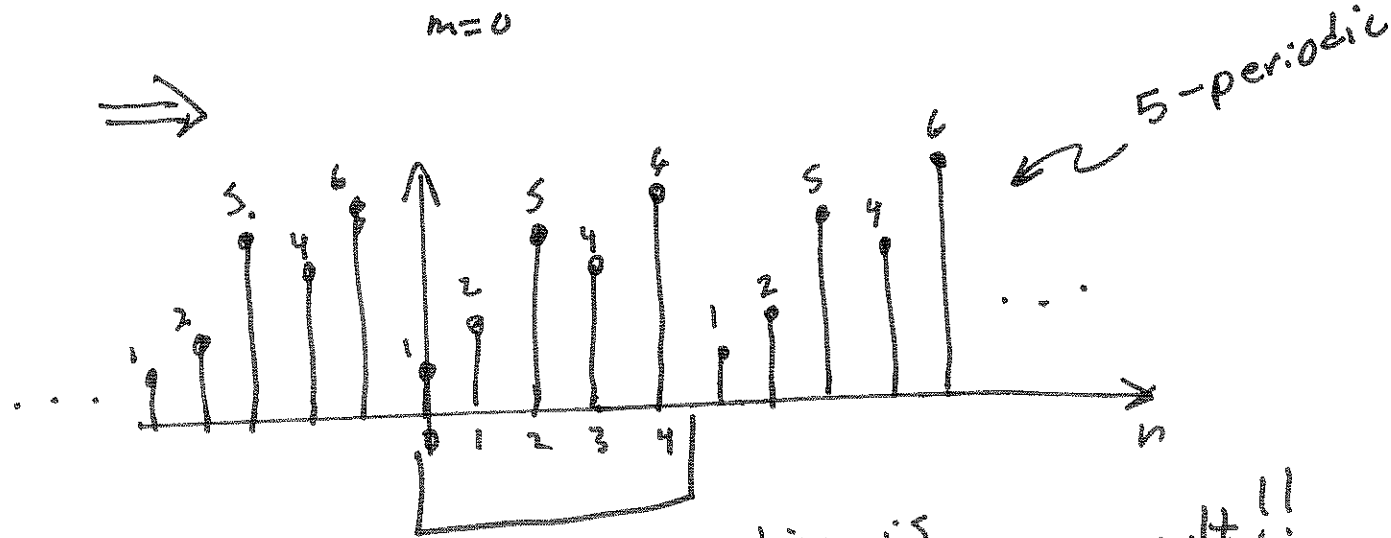
$$h[n] = \{1, 0, 2, 0, 0\}$$

← zero-padded to length-5 signals (5 being the duration of the regular convolution result)

Now take DFTs of zero-padded signals.

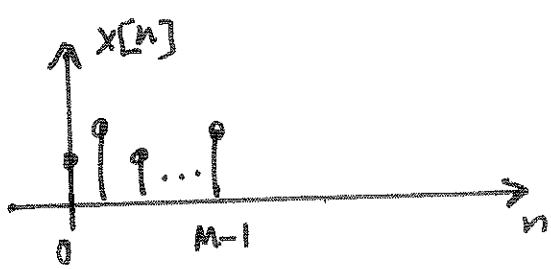
$$\tilde{y}[n] = \frac{1}{N} \sum_{k=0}^{4} X[k] H[k] e^{j2\pi \left(\frac{k}{5}\right) n}$$

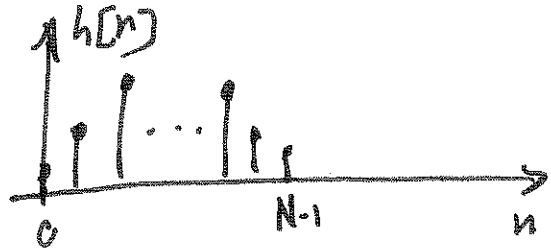
$$= \sum_{m=0}^{4} x[m] \, h[((n-m))_5]$$

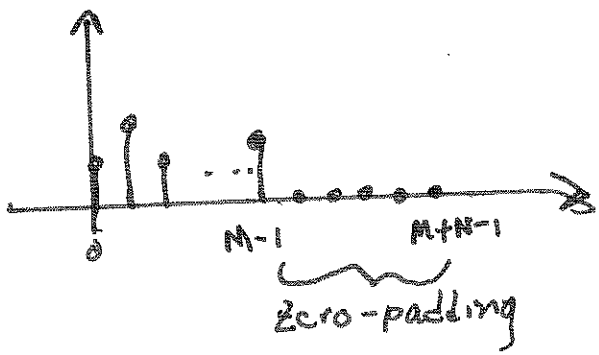


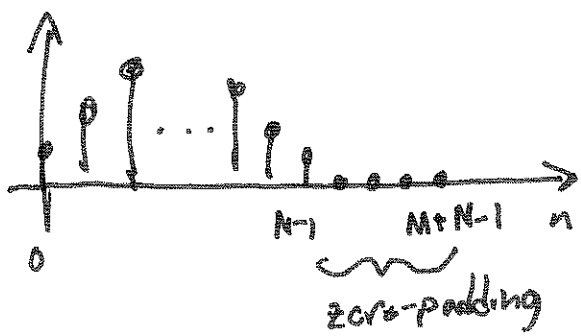
General Result

We can compute the regular convolution result of a convolution of an M-point signal $x[n]$ with an N-point signal $h[n]$ by padding each signal with zeros to obtain two $M+N-1$ length sequences and computing the circular convolution (or equivalently computing the IDFT of $H[k] \cdot X[k]$, the product of the DFTs of the zero-padded signals).

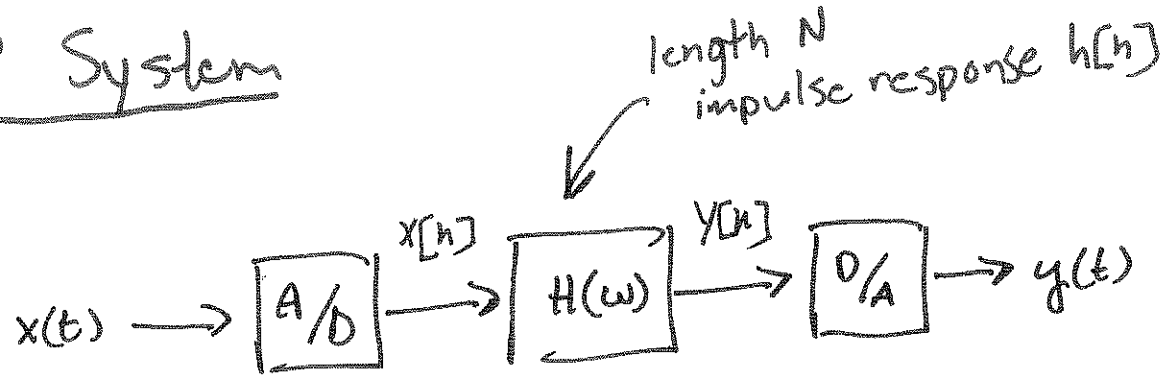








DSP System



(1) Sample finite duration continuous time input $x(t)$ to get $x[n]$, $n = 0, \ldots, N-1$

(2) zero-pad $x[n]$ and $h[n]$ to length $M+N-1$

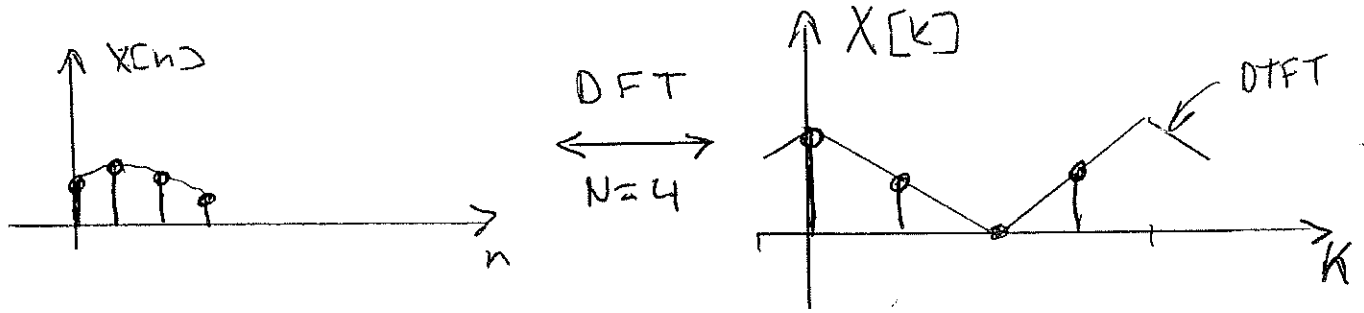
(3) Compute DFTs $X[k]$ and $H[k]$

(4) Compute IDFT of $X[k] \cdot H[k]$

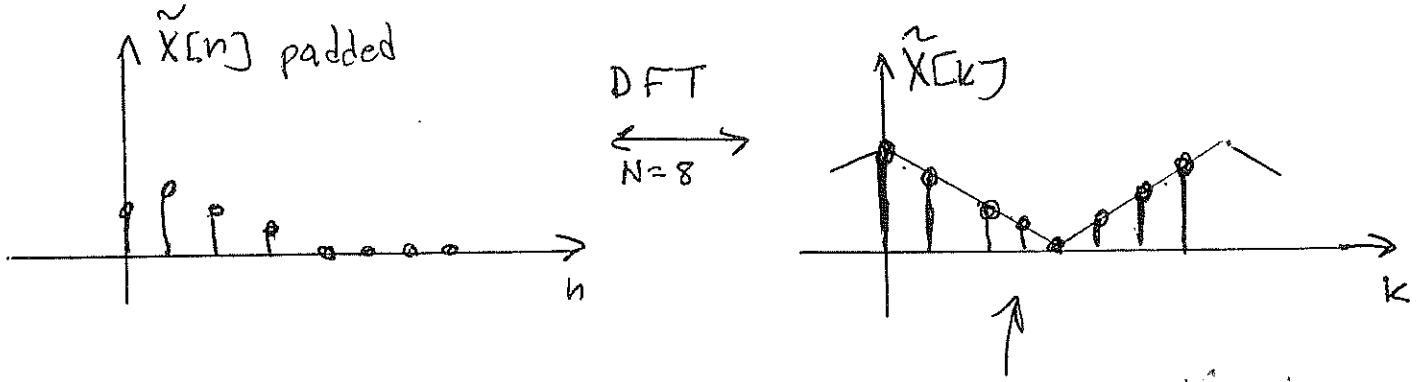
$$y[n] = \tilde{y}[n], \quad n = 0, \ldots, M+N-1$$

(5) Reconstruct $y(t)$

Zero-Padding and DFT Resolution



Zero-padding x[n]



⇒ better "approximation"

Why?

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j 2\pi \left(\frac{k}{N}\right) n}$$

$$\tilde{X}[k] = \sum_{n=0}^{2N-1} x[n]\, e^{-j 2\pi \left(\frac{k}{2N}\right) n} \qquad \left(\text{since } x[n]=0 \text{ for } n=N,\ldots,2N-1\right)$$

$$= \sum_{n=0}^{N-1} x[n]\, e^{-j 2\pi \left(\frac{k}{2N}\right) n}$$

↖ Same form as $X[k]$ except finer sampling in frequency!

Another Perspective: $x[n], h[n]$ = N-point signals

zero-padding in time
= finer sampling in frequency

Define

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j 2\pi \left(\frac{k}{M}\right) n} \qquad k = 0, \ldots, M-1, \quad M \gg N$$

$$H[k] = \sum_{n=0}^{N-1} h[n]\, e^{-j 2\pi \left(\frac{k}{N}\right) n}$$

As $M \to \infty$ our sampling in frequency becomes denser, and in a rough sense

$$X[k], H[k] \rightsquigarrow X(\omega), H(\omega)$$

and

$$X[k] \cdot H[k] \overset{\text{DFT}}{\longleftrightarrow} x[n] * h[n]$$

"approximately"

The FAST FOURIER TRANSFORM
FFT

★ An efficient <u>computational algorithm</u> for computing the DFT

DFT can be <u>expensive</u> to compute <u>directly</u>

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j 2\pi \left(\frac{k}{N}\right) n} \qquad 0 \le k \le N-1$$

For each k, we must execute:

N complex multiplies

$N-1$ complex adds

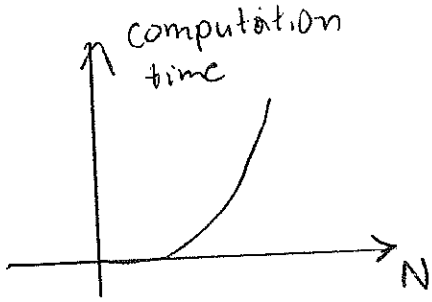
$\Longrightarrow$ Total cost of direct computation of an N-point DFT is

N^2 complex mults

$N(N-1)$ complex adds

How many adds and mults of <u>real</u> #'s ?

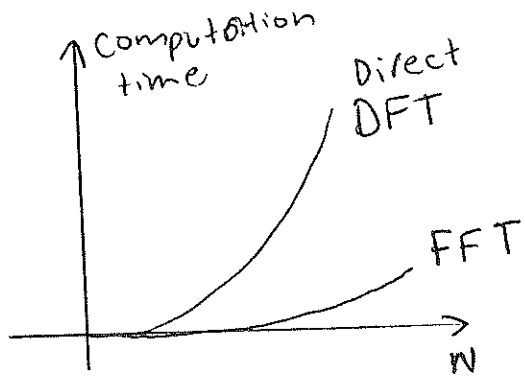
This "$O(N^2)$" (order) computation rapidly gets out of hand, as N gets large:



N	1	10	100	1000	10^6
N^2	1	100	10,000	10^6	10^{12}

The FFT is much more <u>efficient</u> for computing the DFT.

FFT requires only "$O(N \log N)$" computations to compute the N-point DFT



N	10	100	1000	10^6
N^2	100	10,000	10^6	10^{12}
$N \log_{10} N$	10	200	3000	$6 \cdot 10^6$

How long is 10^{12} μsec?

How long is $6 \cdot 10^6$ μsec?

more than 10 days!

- The FFT and digital computers revolutionized DSP (1960-1980)

How does the FFT work?

- The FFT exploits the symmetries of the complex exponentials $\boxed{W_N^{kn} \triangleq e^{-j\frac{2\pi}{N}kn}}$
- W_N^{kn} are called "twiddle factors"

SYMMETRY 1: Complex Conjugate Symmetry

$$\boxed{W_N^{k(N-n)} = W_N^{-kn} = \left(W_N^{kn}\right)^*}$$

$$e^{-j2\pi\left(\frac{k}{N}\right)(N-n)} = e^{+j2\pi\left(\frac{k}{N}\right)n} = \left(e^{-j2\pi\left(\frac{k}{N}\right)n}\right)^*$$

SYMMETRY 2: Periodicity in n & k

$$\boxed{W_N^{kn} = W_N^{k(N+n)} = W_N^{(k+N)n}}$$

$$e^{-j\frac{2\pi}{N}kn} = e^{-j\frac{2\pi}{N}k(N+n)} = e^{-j\frac{2\pi}{N}(k+N)n}$$

$$\boxed{W_N = e^{-j\frac{2\pi}{N}}}$$

DECIMATION IN TIME FFT

- Just one of <u>many</u> different FFT algorithms
- <u>IDEA</u> : Build a DFT out of smaller and smaller DFTs by decomposing $x[n]$ into smaller and smaller subsequences.
- Assume $N = 2^m$ (a power of 2)

DERIVATION

- N is <u>even</u>, so we can compute $X[k]$ by separating $x[n]$ into <u>two</u> subsequences each of length $N/2$.

$x[n]$ → $N/2$ n-even samples
$x[n]$ → $N/2$ n-odd samples

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad 0 \le k \le N-1$$

$$= \sum_{n \text{ even}} x[n] W_N^{kn} + \sum_{n \text{ odd}} x[n] W_N^{kn}$$

$n = 2r$ (for the even sum) $\quad$ $n = 2r+1$ (for the odd sum)

$$0 \le r \le \frac{N}{2} - 1$$

$$X[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r]\, W_N^{2kr} + \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]\, W_N^{(2r+1)k}$$

$$= \sum_{r=0}^{\frac{N}{2}-1} x[2r]\left(W_N^2\right)^{kr} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]\left(W_N^2\right)^{kr}$$

But $W_N^2 = e^{-j\frac{2\pi}{N}2} = e^{-j\frac{2\pi}{N/2}}$

$$= W_{N/2}$$

$$= \sum_{r=0}^{\frac{N}{2}-1} x[2r]\, W_{\frac{N}{2}}^{kr} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]\, W_{N/2}^{kr}$$

$\frac{N}{2}$-point DFT of even samples $G[k]$

$\frac{N}{2}$-point DFT of odd samples $H[k]$

$$\boxed{X[k] = G[k] + W_N^k H[k]} \qquad 0 \le k \le N-1$$

★ Decomposition of an N-point DFT as a sum of 2 $\frac{N}{2}$-point DFTs

- Why would we want to do this?

Because its more efficient!

Recall: Cost to compute an N-point DFT is approximately N^2 complex mults and adds

But decomposition into 2 $\frac{N}{2}$-point DFTs + combination requires only

$$\underbrace{\left(\frac{N}{2}\right)^2}_{\substack{\frac{N}{2}\text{-point DFT} \\ G[k]}} + \underbrace{\left(\frac{N}{2}\right)^2}_{\substack{\frac{N}{2}\text{-point DFT} \\ H[k]}} + \underbrace{N}_{\text{to combine}} = \boxed{\underbrace{\frac{N^2}{2} + N}_{\text{TOTAL}}}$$

Complex mults and adds

---

ex. Savings

$N = 1000$

$N^2 = 10^6$

$$\frac{N^2}{2} + N = \frac{10^6}{2} + 1000 \quad \text{(small comp to 500,000)}$$

$$\approx \frac{10^6}{2}$$

- So why stop here?! Keep decomposing. Break each of the $\frac{N}{2}$-point DFTs into two $\frac{N}{4}$-point DFTs, etc, ...

- We can keep decomposing:

$$\frac{N}{2^1} = \frac{N}{2}, \frac{N}{4}, \frac{N}{8}, \cdots, \frac{N}{2^{m-1}}, \frac{N}{2^m} = 1$$

$m = \log_2 N$ times

- Computational Cost:

N-pt DFT $\longrightarrow$ two $\frac{N}{2}$-pt DFTs

Cost: $N^2 \longrightarrow 2\left(\frac{N}{2}\right)^2 + N$

So, replacing each $\frac{N}{2}$-pt DFT with two $\frac{N}{4}$-pt DFTs will reduce cost to

$$\longrightarrow 2\left[2\left(\frac{N}{4}\right)^2 + \frac{N}{2}\right] + N$$

$$= 4\left(\frac{N}{4}\right)^2 + 2N \qquad \boxed{P=2}$$

$$= \frac{N^2}{2^2} + 2 \cdot N = \frac{N^2}{2^P} + PN$$

As we keep going $p = 3, 4, \ldots, m = \log_2 N$ we get

$$\text{Cost} = \frac{N^2}{2^{\log_2 N}} + N\log_2 N$$

$$= \frac{N^2}{N} + N\log_2 N$$

$$= N + N\log_2 N \qquad \left(\begin{array}{c}\text{total \# of}\\ \text{complex adds}\\ \text{and mults}\end{array}\right)$$

For N large:

$$\underline{\text{Cost}} \approx N\log_2 N \qquad \left(\begin{array}{c}\text{since } N\log_2 N \gg N\\ \text{for large } N\end{array}\right)$$

or

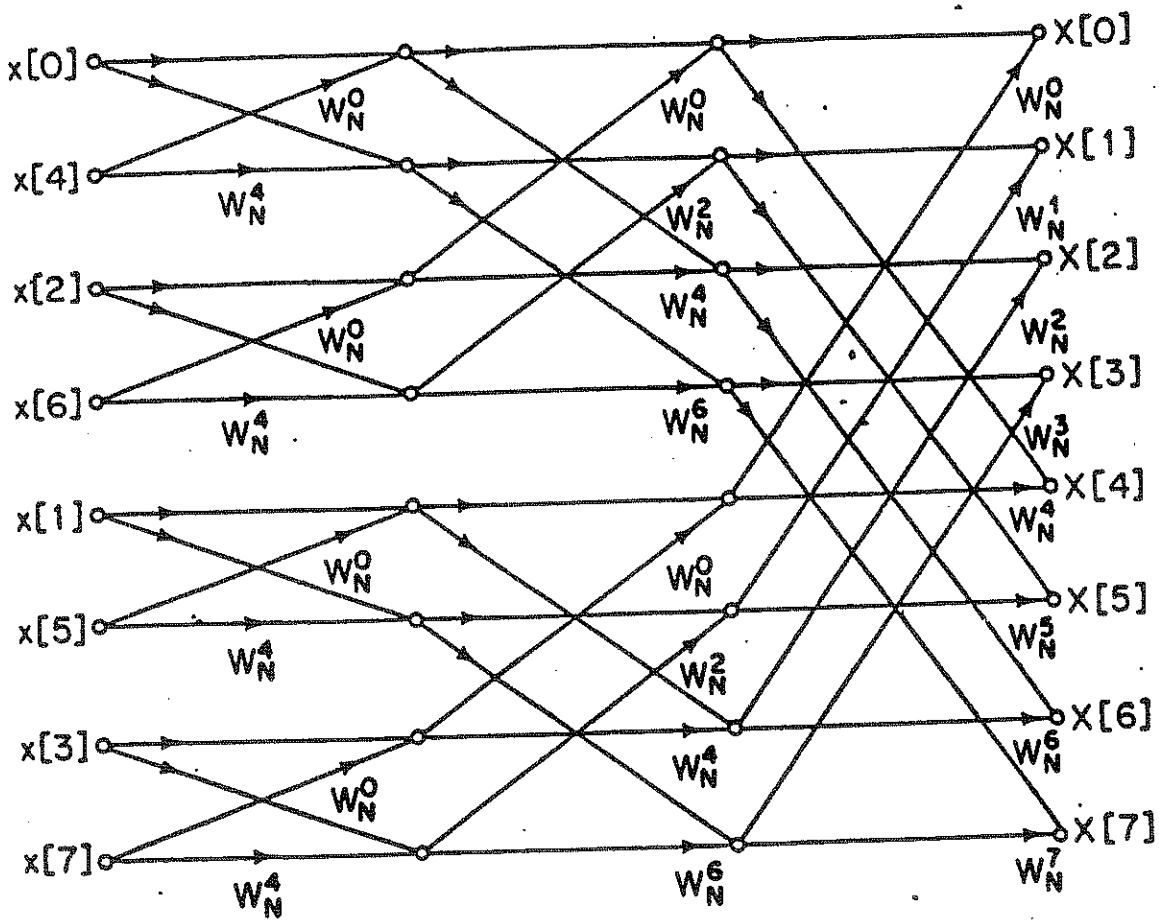
$$\text{"}O(N\log_2 N)\text{"}$$

THE FINAL PICTURE

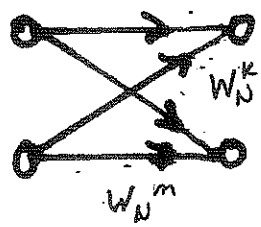
N = 8 point FFT

o Summing nodes

W_N^k twiddle multiplication factors



Note weird order of time samples



called "butterflies"

More advanced FFT topics (see OSB)

→ other FFT algorithms

- decimation in frequency

→ related transforms

DCT - discrete cosine transform
(real-valued sinusoids)

→ non-"power of 2" length FFTs

$N \neq 2^r$ PRIME factor algorithms

Burrus & Parks - 1985

History of the FFT

- dates back to Gauss 1805
(see p. 632 OSB)

- Burrus & Parks led development of FFTs here at Rice in 70's-80's

FAST CONVOLUTION USING THE FFT

* Important application of the FFT *

Question: How many complex multiplies and adds are required to convolve two N-pt. sequences

$$y[n] = \sum_{m=0}^{N-1} x[m]\, h[n-m]$$

There are $2N-1$ non-zero output points and each will be computed using

N complex mults

$N-1$ complex adds

$\longrightarrow$ Total Cost = $(2N-1)(N+N-1) \sim O(N^2)$

(1) Now zero-pad these two sequences to length $2N-1$, take DFT's using the FFT algorithm

$$x[n] \longrightarrow X[k] \quad , \quad h[n] \longrightarrow H[k]$$

Cost: $O\left((2N-1)\log(2N-1)\right) = O(N \log N)$

(2) Multiply DFTs

$$X[k] \cdot H[k]$$

cost: $O(2N-1) = O(N)$

(3) Inverse DFT using FFT

$$X[k] \cdot H[k] \longrightarrow y[n]$$

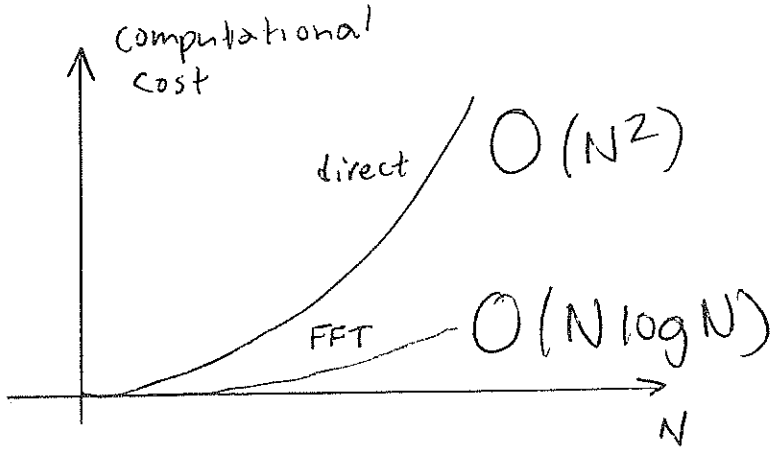
cost: $O((2N-1)\log(2N-1)) = O(N \log N)$

$\Rightarrow$ <u>BOTTOM LINE</u>

Total cost for direct convolution of two N-point sequences $\sim O(N^2)$

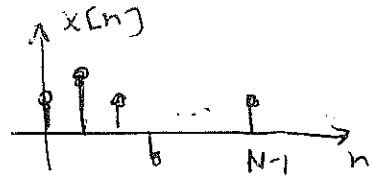
Total cost for convolution using FFT algorithm $\sim O(N \log N)$

HUGE SAVINGS



Summary of DFT

- $x[n]$ is an N-point signal



- $$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}$$ DFT

$$= \sum_{n=0}^{N-1} x[n]\, W_N^{kn} \qquad \left(W_N = e^{-j\frac{2\pi}{N}} \text{ "twiddle factor"} \right)$$

- What is the DFT

$$X[k] = X\left(F = \frac{k}{N}\right) = \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi Fn}$$

$X[k]$: DTFT of $x[n]$

$\sum$: DTFT of $x[n]$ at digital freq. F

(Samples of the DTFT. We can do frequency domain analysis on a computer!)

- Inverse DFT (IDFT)

$$x[n] = \frac{1}{N}\sum_{n=0}^{N-1} X[k]\, e^{j\frac{2\pi}{N}kn}$$

- build $x[n]$ using "simple" complex sinusoidal "building block" signals
- amplitude of each complex sinusoidal building block in $x[n]$ is $\frac{1}{N}X[k]$.

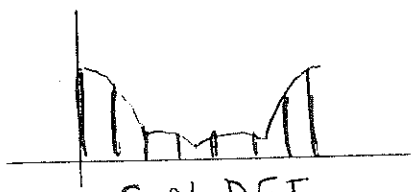
- ## Circular Convolution

$$x[n] \textcircled{N} h[n] \xleftrightarrow{DFT} X[k] \cdot H[k]$$

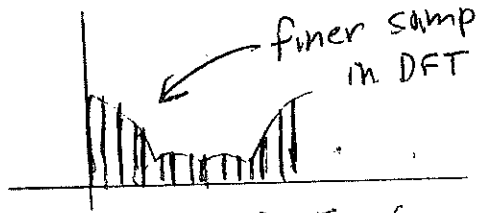
- ## Regular Convolution from Circular Conv.

  - zero pad $x[n]$ and $h[n]$ to length = length(x) + length(h) − 1

  - zero padding increases frequency resolution in DFT domain



8-pt DFT of 8-pt signal



16-pt DFT of same signal padded with 8 additional zeros

- ## The Fast Fourier Transform (FFT)

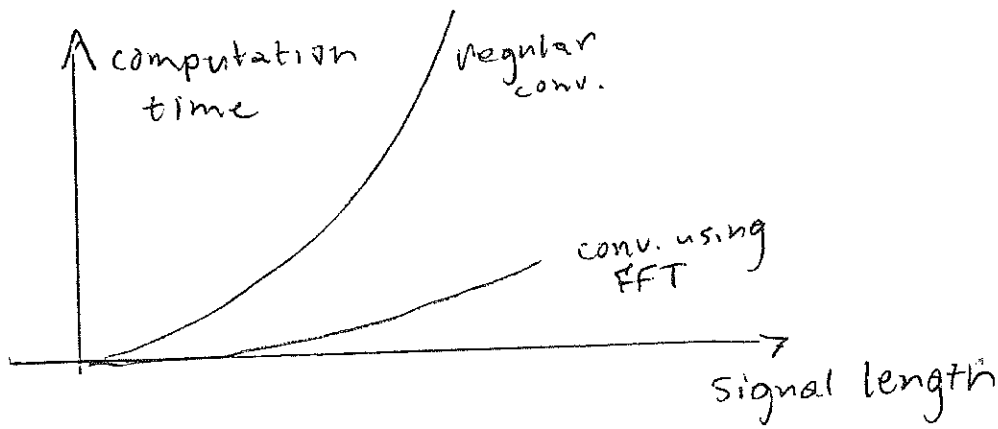
  - efficient computational algorithm for calculating the DFT
  - "divide and conquer"
  - break signal into even and odd samples, keep taking shorter and shorter DFTs, then build N-pt. DFT by cleverly combining shorter DFTs.

N-pt DFT $O(N^2) \searrow O(N \log_2 N)$

Fast Convolution

- use FFT's to compute circular convolution of zero padded signals
- much faster than regular convolution if signal lengths are long

$$O(N^2) \searrow O(N \log_2 N)$$



DFT AS A MATRIX OPERATION

Recall:

Vectors in $\mathbb{R}^N$: $\underline{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$, $x_i \in \mathbb{R}$

Vectors in $\mathbb{C}^N$: $\underline{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$, $x_i \in \mathbb{C}$

Transposition:

$$\underline{x}^T = [\, x_0 \; x_1 \; \cdots \; x_{N-1} \,]$$

$$\underline{x}^H = [\, x_0^* \; x_1^* \; \cdots \; x_{N-1}^* \,]$$ (transpose and conjugate)

Inner product:

real: $\underline{x}^T \underline{y} = \sum_{i=0}^{N-1} x_i y_i$

complex: $\underline{x}^H \underline{y} = \sum_{i=0}^{N-1} x_n^* y_n$

Matrix Multiplication:

$$\underline{A}\,\underline{x} = \begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0,N-1} \\ a_{10} & a_{11} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{N-1,0} & a_{N-1,1} & \cdots & a_{N-1,N-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix}$$

$$y_k = \sum_{n=0}^{N-1} a_{kn}\, x_n$$

Matrix Transposition:

$$\underline{A}^T = \begin{bmatrix} a_{00} & a_{10} & \cdots & a_{N-1,0} \\ a_{01} & a_{11} & \cdots & a_{N-1,1} \\ \vdots & \vdots & & \vdots \\ a_{0,N-1} & a_{1,N-1} & \cdots & a_{N-1,N-1} \end{bmatrix}$$

Swap rows with columns

Hermitian transpose

$$\underline{A}^H = (\underline{A}^T)^*$$

$$(\underline{A}^T)_{kn} = \underline{A}_{nk}$$

$$(\underline{A}^H)_{kn} = \underline{A}^*_{nk}$$

Now let's represent the DFT in vector-matrix notation.

$$\underline{x} = \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$ ← vector of time samples

$$\underline{X} = \begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix} \in \mathbb{C}^N$$ ← vector of DFT coefficients

How are $\underline{x}$ and $\underline{X}$ related:

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}$$

$$a_{kn} = \left(e^{-j\frac{2\pi}{N}}\right)^{kn} = W_N^{kn}$$

$$\Rightarrow \quad \underline{X} = \underline{W}\,\underline{x}$$

$\underline{X}$: DFT vector; $\underline{W}$: matrix; $\underline{x}$: time domain vector

$$\left[\underline{W}\right]_{kn} = \left(e^{-j\frac{2\pi}{N}}\right)^{kn}$$

$$\underline{X} = \underbrace{\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}}_{\underline{W}} \begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix}$$

IDFT:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \underbrace{\left(e^{j\frac{2\pi}{N}}\right)^{nk}}_{(W_N^{nk})^*}$$

← matrix Hermitian transpose

$$\Rightarrow$$

$$\underline{x} = \frac{1}{N} \underbrace{\underline{W}^H}_{\text{inverse DFT matrix}} \underline{X}$$

↑ time Vector ↑ DFT Vector